TH[illegible]ING COMPUTER PROJECTS MAGAZINE APRIL 1985

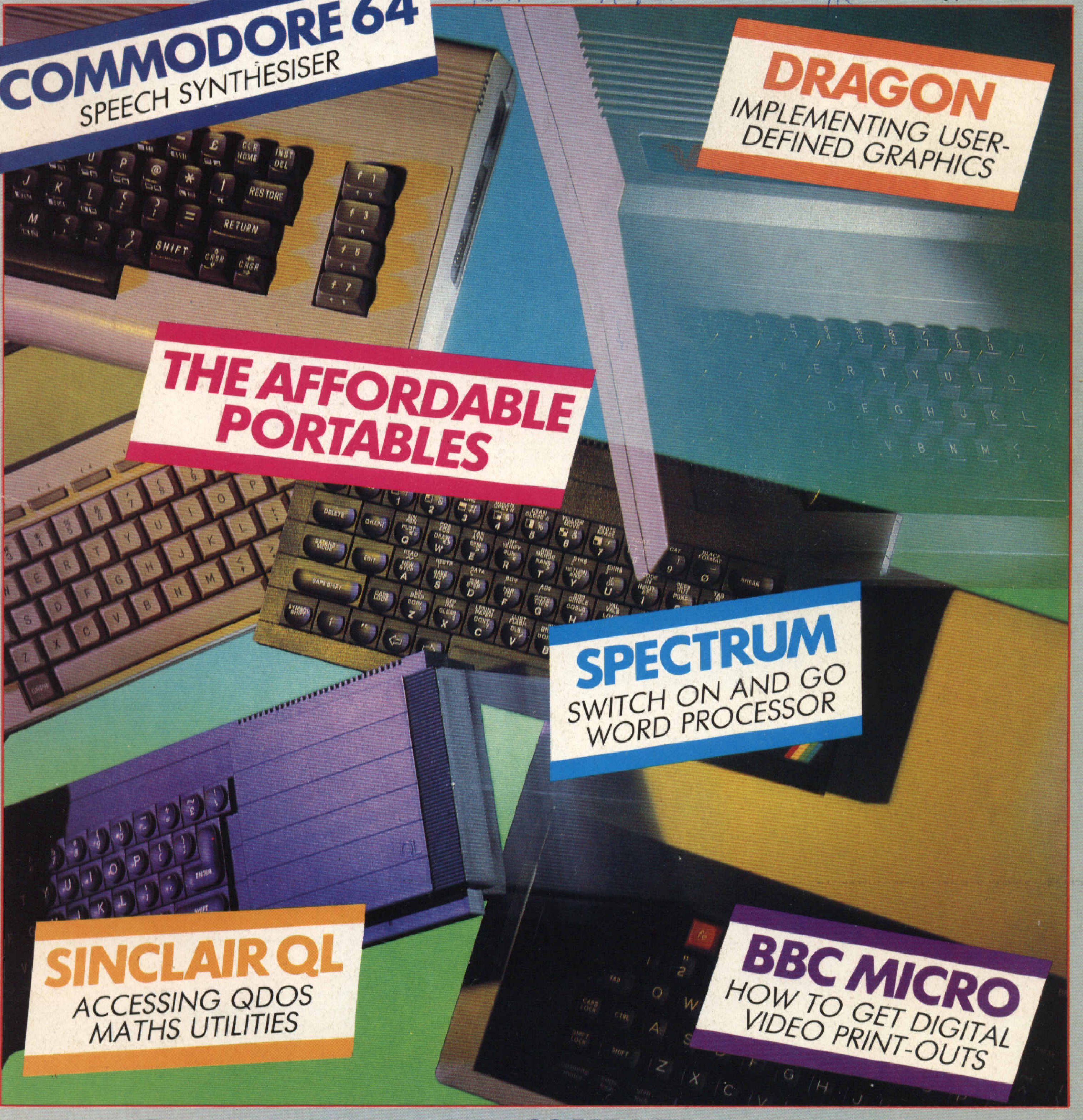
4 DRAGON COMPUTERS TO BE WON

[illegible]LECTRONICS & COMPUTING

AN EMAP PUBLICATION

USA $2.95
Germany D6.00
Singapore S$4.95
95p

22 33 38 ULA 59 63 Book
48 SPEC LIGHT PEN o/k



COMMODORE 64
SPEECH SYNTHESISER

DRAGON
IMPLEMENTING USER-DEFINED GRAPHICS

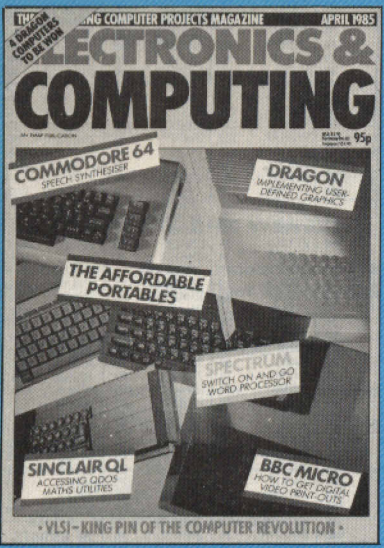
THE AFFORDABLE PORTABLES

SPECTRUM
SWITCH ON AND GO WORD PROCESSOR

SINCLAIR QL
ACCESSING QDOS MATHS UTILITIES

BBC MICRO
HOW TO GET DIGITAL VIDEO PRINT-OUTS

• VLSI – KING PIN OF THE COMPUTER REVOLUTION •



COVER PHOTO BY ROB BRIMSON

Electronics & Computing Monthly
Priory Court, 30-32 Farringdon Lane,
London, EC1R 3AU

Editorial 01-251-6222
Editor Gary Evans
Deputy Editor William Owen
Production Editor Liz Gregory

Advertising 01-251-6222
Advertisement Manager Anthony Herman
Advertising Executive Tracy Keighley
Advertising Production Serena Hadley

Production 01-251-6222
Art Editor Jeremy Webb
Make-up Time Graphics
Publisher Terry Pratt

Distribution
EMAP National Publications

Published by
EMAP Business and
Computer Publications

Printed by
Riverside Press, England

Subscriptions
Subscriptions and back issues please telephone 01-251 6222 for details

Electronics & Computing Monthly is normally published on the 13th day of each month.

 Subscription rates: UK £15.00 incl. post. For overseas rates apply to Subscription Dept., Competition House, Farndon Road, Market Harborough, Leicestershire. Back issues available from: EMAP National Publications (*E&CM* Back Numbers), Bretton Court, Peterborough, PE3 8DZ. Phone: 0733 264666.

ABC
MEMBER OF THE AUDIT BUREAU OF CIRCULATIONS

ELECTRONICS & COMPUTING Contents

Vol. 5 Issue 4

PROJECTS

FEATURES

REVIEWS

PLUS

We apologise to readers for the fact that a number of our regular features have had to be held over until next month due to the pressure on editorial space.

See page 40 for details of this month's competition with four Dragon computers to be won.

New Atari 68000 machine will run GEM software

The Sinclair QL will face severe competition later this year when two new machines, the Commodore 128 and Atari ST series, are launched in the UK.

Both machines saw the light of day at CES Las Vegas in January, and both are attempting to follow the QL into that indeterminate market between the home and business user by providing a low-cost but powerful computer capable of running games and business software alike.

The Atari ST range is a significant step forward in low-cost computing. It is a 68000-based computer which will be available in two versions, the 130ST and 520ST. The processor is similar to the 68008 used in the QL, which is a cut-down version of the 32-bit 68000 with an 8-bit rather than 16-bit bus.

Both versions of the ST run GEM (Graphics Environment Manager); this is Digital Research's menu and ikon driven operating system, similar to that used by Apple on the Macintosh and far in advance of QL QDOS.

The main difference between the 130 and 520 is the available user memory. The 130 has 128K RAM: the 520 has 512K RAM. Each has 192K ROM expandable to 320K, RS232 and Centronics interfaces; video, RGB and TV outputs (with bit-mapped display) and joystick ports.

An Atari spokeswoman in the US told *E&CM* that both machines would be on sale in the US in March or April. Atari UK expect that the British launch will be in May or June. No statement on price was forthcoming, but it is likely to be in the $300 to $400 range.

Commodore were equally vague about the UK launch date of the C128: 'Certainly before the end of this year, logically mid to end summer, in time for the Christmas market'.

The C128 is an 8-bit machine and a bit of a mongrel at that. It has no less than three processors: a 6510 (to emulate the Commodore 64); and 8502 (the C128's 'own' processor) and a Z80 to run CP/M programs.

Different amounts of RAM are available depending on which processor is in use: 64K on the 6510, and 128K on the 8502 and Z80. The main function of the 8502 is fast data transfer between the computer and Commodore's 1571 disk drive.

A mass of software will be available to users of the C128. All CBM64 programs can run on this machine, as well as standard CP/M packages. So there will be games, wordprocessors and spreadsheets galore.

But in order to compete in the UK market the C128 will have to be competitively priced – there are a lot of cheap CP/M machines around, which may not run CBM64 games but most business users don't need to. When Commodore release a true 16-bit machine, as expected, then they will be able to stand up to the Atari STs and other low-cost home/business computers. *E&CM* will benchtest both the Atari and Commodore machines as soon as they can be obtained from the US.

● Commodore have slashed the price of the Plus/4 from £300 to £150. This almost certainly means that the less powerful C16, priced at £130, will be phased out, and although the CBM64 may retain its popularity Commodore can expect the Plus/4 with its integral software to take a leading position in the market. Following price cuts Commodore's UK general manager, Howard Stanworth, resigned his position. In the first six months of the current financial year the profits of Commodore UK were halved.

A commendably clear Act

The confusion surrounding the law of copyright in relation to computer software could be ended if a bill currently working its way through Parliament receives the Royal Assent. The Copyright (Computer Software) Amendment Bill aims to bring computer software within the scope of the Copyright Act of 1956. This will mean that computer software can be protected under the same well established laws that relate to, for example, audio recordings or video tapes.

The Bill is quite explicit in what it sets out to achieve. The first section states that the Copyright Act shall apply "in relation to a computer program as it applies to a literary work". The 1956 Act has powerful 'teeth' in that there is a provision to fine anyone found to be in breach of the Act a sum of £2000 for each infringement or to impose a two month prison penalty.

E&CM welcomes the new Act and salutes the work of FACT – the Federation Against Software Theft, for campaigning to bring about legislation which will give a clear cut legal remedy to any software author who finds himself the victim of piracy. The example of the success that the video trade have had in combating piracy using the 1956 act as its major weapon should give software publishers hopes that they too will achieve similar levels of success.

Despite the commendably clear phrasing of the Bill which should provide a straightforward legal recourse in cases of simple 'back to back' copying of software, there are almost certain to be some test cases in instances when breach of copyright is less clear. The amendment states that 'for the purposes of the said Act of 1956 in relation to a computer program, a version of the program in which it is converted into or out of a computer language or code, or into a different computer language or code, is an adaption of the program'. The legal meaning of the word adaption will be taken to be that established by the 1956 Act in relation to literary works. Quite how wide this section of the Act is seen to be will only be established as and when it is called into play. For example, could the author of a specialised accounting package implemented on a specific computer claim that no other person could produce an application package for another system as this would be in breach of his copyright? Only time and the due process of law will provide clarification on the wider implications of the amendment to the law of copyright.

Gary Evans

Maplin fires broadside against piracy

Software companies have long been searching for a means to prevent their programs being copied (pirated) either by individuals or by organised (criminal) gangs. The unauthorised duplication of software is claimed to be costing the industry anything between £30 and £100m each year. 'Safe Load' is a recently launched protection system that could prove to be the solution to the problem of computer software piracy.

The 'Safe Load' system was developed by Keith Halliwell and will be marketed by Maplin Electronic Supplies. 'Safe Load' requires no hardware modification to either the computer or the cassette recorder used to load data but the 'Safe Load' data cassettes themselves are of a special design. Outwardly the cassettes are identical to a standard audio cassette but there are differences to the internal design of a 'Safe Load' cassette. The computer detects these differences and when the Safe Load protected software is copied onto a standard cassette, the software will still load as usual but the check for the presence of a 'Safe Load' cassette will fail and the program will not run. This is the secret of the 'Safe Load' system, it does not seek to prevent the audio copying of software by low level recording techniques, something which has proved unsuccessful in the past, but relies on the special cassette housing. Thus the system allows perfect audio copies to be made but prevents such copies from being used.

At the launch of the system a Safe Load protected program was copied to an ordinary cassette tape and this copy did indeed fail to load. A copy of the copy was then recorded on a blank 'Safe Load' tape and this third generation tape was loaded and then run without any difficulty.

At the launch of the system Maplin stated that they would police the system by way of random checks throughout the country and that they would not hesitate to take legal action against any individual or group of people infringing their patent in respect of the system.

As yet there is no software available with the 'Safe Load' anti copy system incorporated but many well known software publishers are bound to show a keen interest in a system that could once and for all rid the industry of the unwelcome attentions of the software pirate.

Revitalised Acorn may spurn home market

Acorn have been saved by Olivetti from an already bursting computer companies' graveyard with a package that will 'reinforce Acorn's position in educational and allied markets' but 'further reduce Acorn's dependence on the volatile home computer market'. This may mean the death of the Electron.

The immediate cause of the cash crisis which led to a partial takeover of Acorn Computers by Olivetti was poor pre-Christmas sales of the Electron.

Sixty five per cent of Acorn's £10.9 million half year loss was accounted for by reducing the value of unsold machines stored in warehouses, cancelled orders, and making refunds to the trade because of forced price cuts on the Electron.

Production of both the BBC B and Electron has ceased, according to a spokesman at Acorn's Public Relations company. It is not clear whether this is a temporary or permanent suspension but BBC, if not Electron users, can probably assume that their machines will continue to be supported with new hardware and software. 'The company' says the spokesman, 'is back on the straight and narrow'.

Immediately after Christmas Acorn discontinued its US operation after losses of £6 million, and cut back its dealer network in the UK. Rumours of poor sales caused Acorn share prices to fall in the week preceding 22 January, when Chris Curry and Herman Hauser, joint founders of Acorn, announced a price cut of £70 off the Electron and trade-in deal of £50 discount on a BBC Micro.

Trading in Acorn shares was suspended on February 6 following the departure of the company's merchant bank and stockbrokers, and what Chris Curry described as 'inaccurate and uninformed' press reports as share prices plummetted to 28p from a high of 193p.

Acorn refused to make any statement about its plans for reorganisation and there was speculation that production of the Electron had ceased and of takeover bids by Sinclair Research and Thorn EMI. On 14 February at the request of circuit board suppliers Circuit Techniques, who were owed £19,000, a winding-up order was issued to Acorn by the High Court. It was at this point that Olivetti stepped in with an £18 million rescue package.

The deal, which splits Acorn into four sectors – Business, Scientific and Industrial, Home, and Education – gives Olivetti a 49.3% stake in the company worth £10.4 million. Barclays Bank has doubled its loan facilities to £12 million, and major creditors have agreed to accept delayed repayments. Curry and Hauser lose control of the company and 'have gone into a back seat' with a stake reduced from 85% to 36.5%.

What this has cost users of Acorn computers remains to be seen. No detailed statement has been issued regarding future support in the home and education sectors, or the fate of Acornsoft, or of new projects such as the 'C' and upgrades to the BBC. AB Electronics may resume production of computers now that it is satisfied that Acorn are financially viable – stocks of the BBC Model B with DFS fitted are already exhausted.

Micro floppy disc interface discovered on Spectrum

The Spectrum continues to expand, now with a 3.5" single disk drive module built by Opus Supplies Ltd, called the Discovery 1, and priced at £199.95.

Opus has also announced the Discovery Plus, an upgrade package for conversion to dual disc drives, priced at £139.95, and the Discovery 2, a purpose-built dual drive system priced at £325.95.

Discovery 1 will be exclusively marketed by Boots from early March. The module sits at the back of the Spectrum, bolted to the rear edge connector, and includes a peripheral through connector, Centronics printer interface, joystick interface, mono video output, and an onboard power supply.

The single sided, 40 track, double-density 3.5" drive has a formatted capacity of 180Kbytes. The system supports Random Access files and has firmware to implement RAM disks. It is fully compatible with Microdrive/Interface 1 syntax, and has all Microdrive commands plus extensions, so is compatible with all microdrive software.

Software will be available immediately on 3.5" disks. Six suppliers have agreed to distribute mostly games, but also utilities and wordprocessing and spreadsheet software.

PCW makes takeover bid for Channel 4

We had hoped to extend congratulations to Channel 4 and the '4 Computer Buffs' production team for their work in bringing to the TV screens a programme that was, in its pre-transmission publicity, described as a programme aimed at the more advanced computer user. In the event however judging on the evidence of the first program, the only one to have been broadcast at the time of writing, we feel that the programme should be accompanied by a warning decreeing that too much PCW is dangerous for your health. It was surprising to see a commercial TV company associate itself so closely with one particular publication.

It was easy to loose track of the number of mentions that the March issue of PCW was given in the first '4 Computer Buffs' though. The bench tests were developed from PCW's and the item was discussed with two of the magazine's staff and two of its regular freelance writers. Further details of the 'photon' software project and the 'Trundle' robot were to be found in PCW. The news/gossip section that rounded off the programme was also presented by a well known PCW face.

While not wishing to appear churlish, '4 Computer Buffs' will not make many friends in other sections of the extensive computer media industry while it so closely identifies itself with just one of the hundreds of publications dealing with computer related subjects.

Hardware and software guides to BBC Micro

Elsewhere in this issue you will find an article from Mike James in which he presents a detailed guide to the operation of the BBC microcomputer from a hardware point of view. While Mike manages to pack a great deal of information into the limited space allowed to him, readers requiring further detail of the circuit may be interested in Wise Owl Publications' Hardware Guide for the BBC Microcomputer'.

This extensive guide provides a full description of the micro's circuitry, some tips on how some common generic problems may be overcome and, for the more advanced user, some fault finding tips and a full set of manufacturers' data sheets describing the main components of the computer.

The book is comprehensively illustrated both with diagrams and fully annoted pictures. Wise Owl Publications are at the Hull Innovation Centre, Guildhall Road, Queens Gardens, Kingston-Upon-Hull.

Another book of interest to owners of the BBC micro and also of Computer Concepts' 'Wordwise' word processor is the 'Wordwise Applications Guide' by Paul Beverley. The guide offers help for beginners and experienced users alike and provides solutions to many of the most common problems users have with Wordwise.

The guide costs £6.50 + 50p p&p and is distributed by TER Roberts, Lamorna, The Street, Bunwell, Norfolk, NR16 1NA.

VIDEO PRINTER

John Yau's project allows a BBC micro to grab a video frame and to produce poster sized prints from the picture information.

The hardware described here will enable a still video frame to be digitised from a standard composite video source (such as a video recorder or camera) and stored onto disk. Although such a process has many general applications the software presented aims to produce digitised picture posters. A BBC Micro fitted with a standard Acorn disk system is required to interface to the digitiser, along with an Epson MX/FX dot matrix printer for the poster printing. Rather than displaying a digitised picture on the microcomputer's screen and then doing a screen dump onto a printer, the technique adopted in this article enables pictures to be printed at far greater resolution than any of the BBC Micro's screen modes can handle. The extensive software presented to support the latter application, along with the minimum component design of the digitiser hardware, makes this a cheap but very effective method of producing those so called 'computerised' pictures.

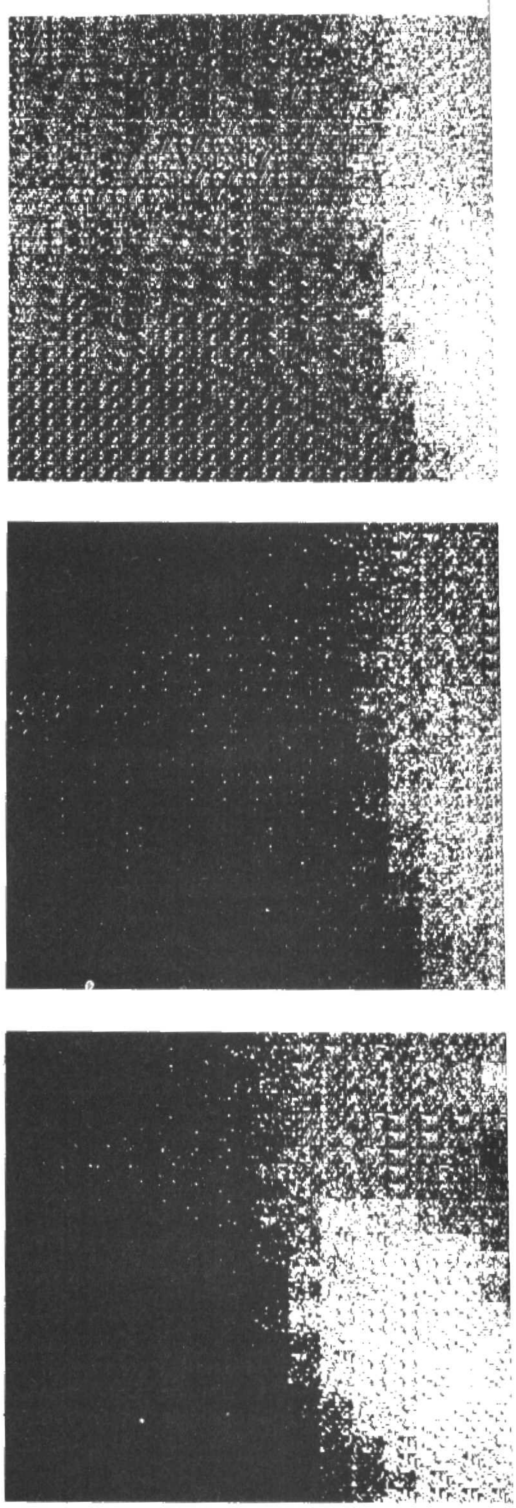
The hardware

Most commercial frame stores consist of a substantial amount of dedicated hardware, with their own microprocessor and large amounts of RAM storage for storing the frame. The latter, along with the utilisation of an expensive 'flash' A/D convertor enables a frame to be snatched in real time, ie in 1/50th sec. For such a system the complexity of the hardware results in costs of over £1000!

The technique adopted here relaxes demands on fast memory access and A/D conversion speeds, resulting in much simpler hardware that costs less than £30 to construct. The method uses the BBC Micro's RAM for storage and digitises over a number of successive frames. Although obviously not as ideal as 'snatching' a single frame, the results are still excellent for the generation of digitised pictures.

Digitising a video picture

A standard TV picture consists of 625 horizontal lines and is transmitted as two consecutive interlaced field frames of 312.5 lines at a field rate of 50Hz. Each line lasts for $64\mu S$ and is followed by a negative going line sync pulse lasting $4.7\mu S$. Between each frame there is a sequence of pulses used for frame synchronisation, consisting of 5 equalising pulses at $32\mu S$ intervals, 5 broad pulses at $32\mu S$ intervals, followed by 5 more equalising pulses (see **Figure 1**). These sync pulses are easy to detect because their voltages dip under the nominal black level below which the picture information part of the video signal does not cross.



Example output illustrating the effect of the software

For each detected field frame a vertical picture slice is digitised (see **Figure 2**). A programmable counter delay is initiated by a line sync pulse. When the delay times out it fires a start conversion pulse to the A/D convertor. The subsequent data is read by the computer within the $64\mu S$ line period. When the end of the line is reached the line sync initiates the same delay again, this time the picture element immediately

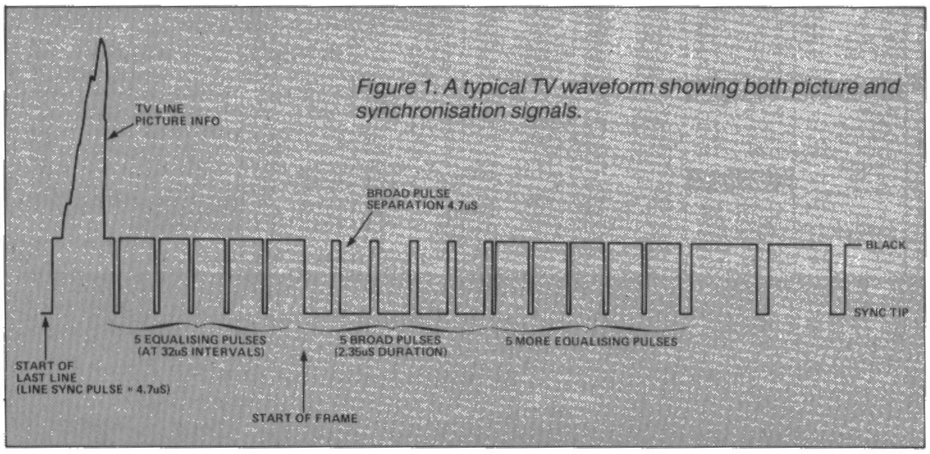
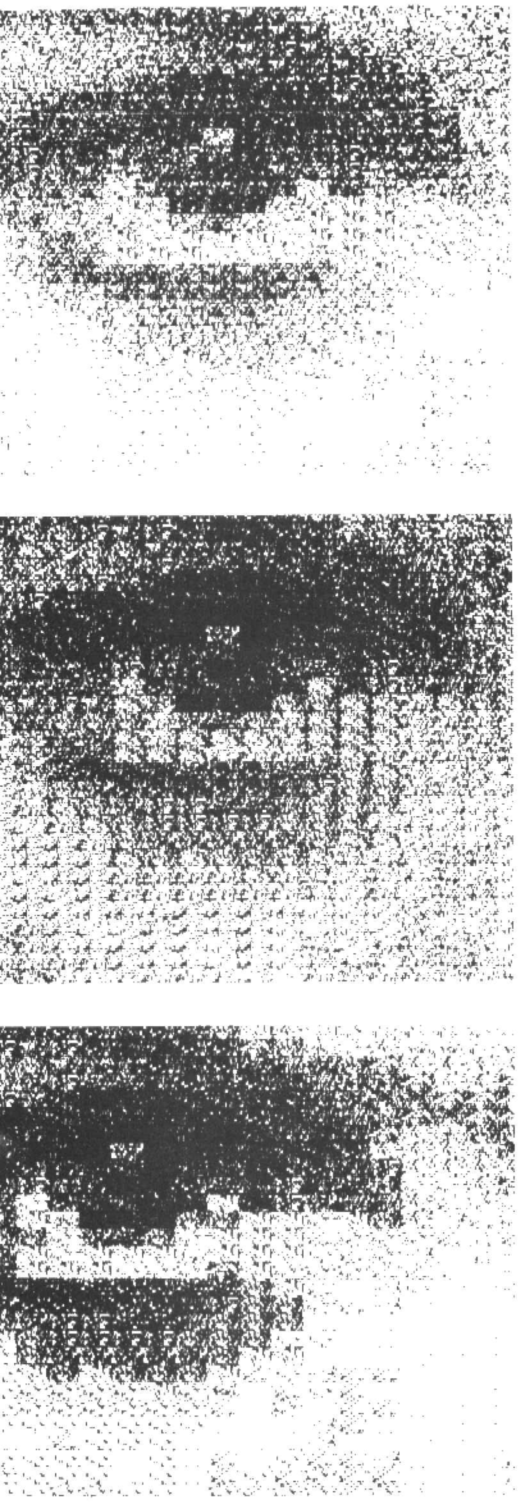


Figure 1. A typical TV waveform showing both picture and synchronisation signals.



contrast control to be described next month.

below the previously digitised one is converted. Hence after one field frame a vertical slice of the picture has been converted.

To capture the next adjacent picture slice the delay is incremented by one picture element time period and the process repeated. The horizontal resolution of the digitiser is fixed at 192, which means that the picture will have to be captured over 384 successive frames ie over a period of about 8 seconds, during which time the picture must be absolutely still. The capture period takes twice the expected time because only every second frame is examined (an adjacent frame is different because of the interlacing).

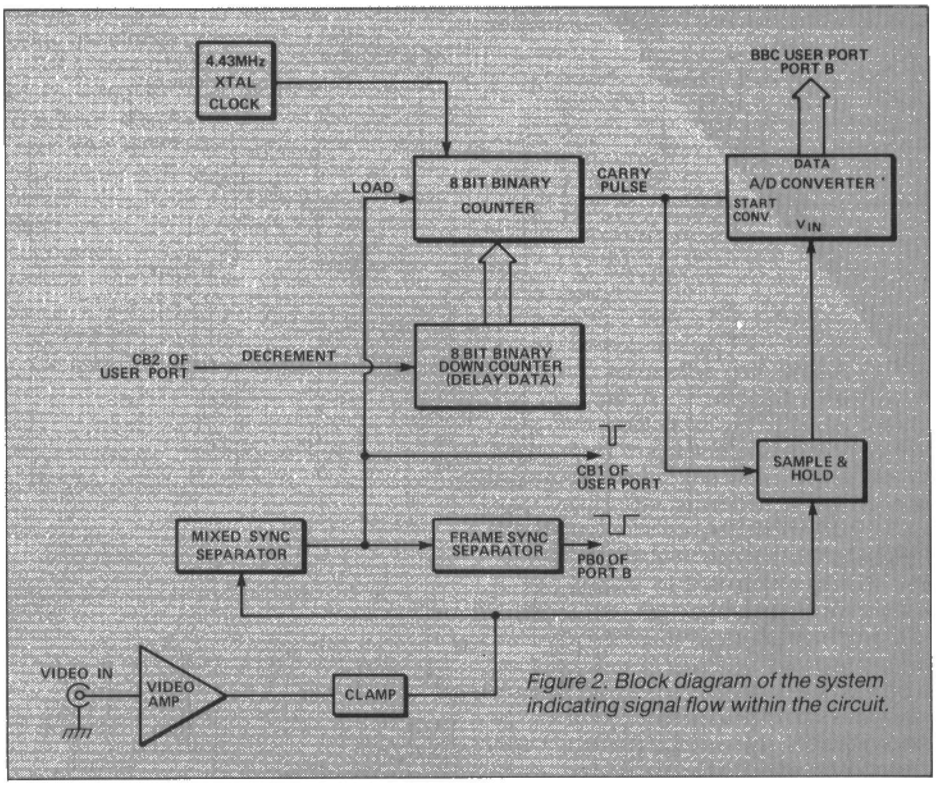


Figure 2. Block diagram of the system indicating signal flow within the circuit.

Specifications

The circuit and software presented in this project gives a resolution of 192 x 200 pixels with 128 brightness levels for each pixel. Almost any standard 1V pk-pk composite video signal will be accepted by the circuit. Although a video camera can be used the video output from a video recorder is by far the best source to work on. With a VCR, pictures that are otherwise moving can be still paused for the 8 seconds required for the frame capture.

To store a picture at the full resolution the required amount of memory is 38 400 bytes (assuming one byte for each pixel), an amount of storage exceeding that of the BBC micro's capacity! The memory problem is solved by software, the trick being to capture the frame in two halves, using 19 200 bytes of the BBC's memory as a buffer. In either case the captured picture should end up as a 38 400 byte file on disk.

Circuit description

The circuit diagram for the video digitiser is shown in **Figure 3.** The composite video signal is buffered by TQ1 feeding a simple tuned circuit notch filter which is adjusted to remove any chroma component in the signal. The signal is then amplified by TQ2,TQ3 and the inverted signal's sync tips are clamped to the DC level set by V2R. Although gated black level clamping is normally adopted, the much simpler sync level clamping was found to be adequate.

The comparator circuit centered around IC1, an LM311, is adjusted so that frame and line syncs are detected. The video signal is low pass filtered with a cut-off of 0.5MHz to ensure that the last of the chroma and high frequency noise is removed before presentation to IC1. A certain amount of hysteresis is also added to IC1 to guard against noisy inputs. The mixed sync output is directed to CB1 of the BBC User Port, the frame sync detector and the load lines of line delay binary counters IC3, IC4.

Frame sync detection is achieved by the integrating network R16, C8 feeding IC2, an NE555 configured as a monostable. The latter triggers during the broad pulse region of the frame sync sequence. This pulse is presented to PB0 to be detected by software. Since the capture process starts after the rising edge of the latter pulse is detected, V4R determines the first line to be digitised.

At the start of the capture process the binary down counters formed by IC5 and IC6 is set to 255 by SW1 (reset). The frame sync pulse is then detected by software. Upon the rising edge of the pulse the BBC User VIA is programmed so that CB1 interrupts the BBC Micro. An on-coming line sync pulse loads the counter with the delay byte from IC5, IC6 and the count commences, as clocked by the 4.43MHz crystal oscillator circuit built around IC9. When the delay expires a short carry pulse is generated by IC4 of the counter circuit. This pulse provides a 220 nS start conversion pulse for the A/D convertor IC8 and also closes the CMOS sample and hold switch IC10 for the same amount of time. It was found by experiment that the best results were obtained using just the CMOS switch capacitance feeding the high impedance of the FET op-amp IC7.

The A/D convertor chosen is the ZN448, a successive approximation device having a conversion time of 10μS, well within the required conversion rate of once every line period (64μS). The device is configured in the circuit to vary to the bipolar mode, so that it converts input voltages within the range –5V to +5V. However only half this

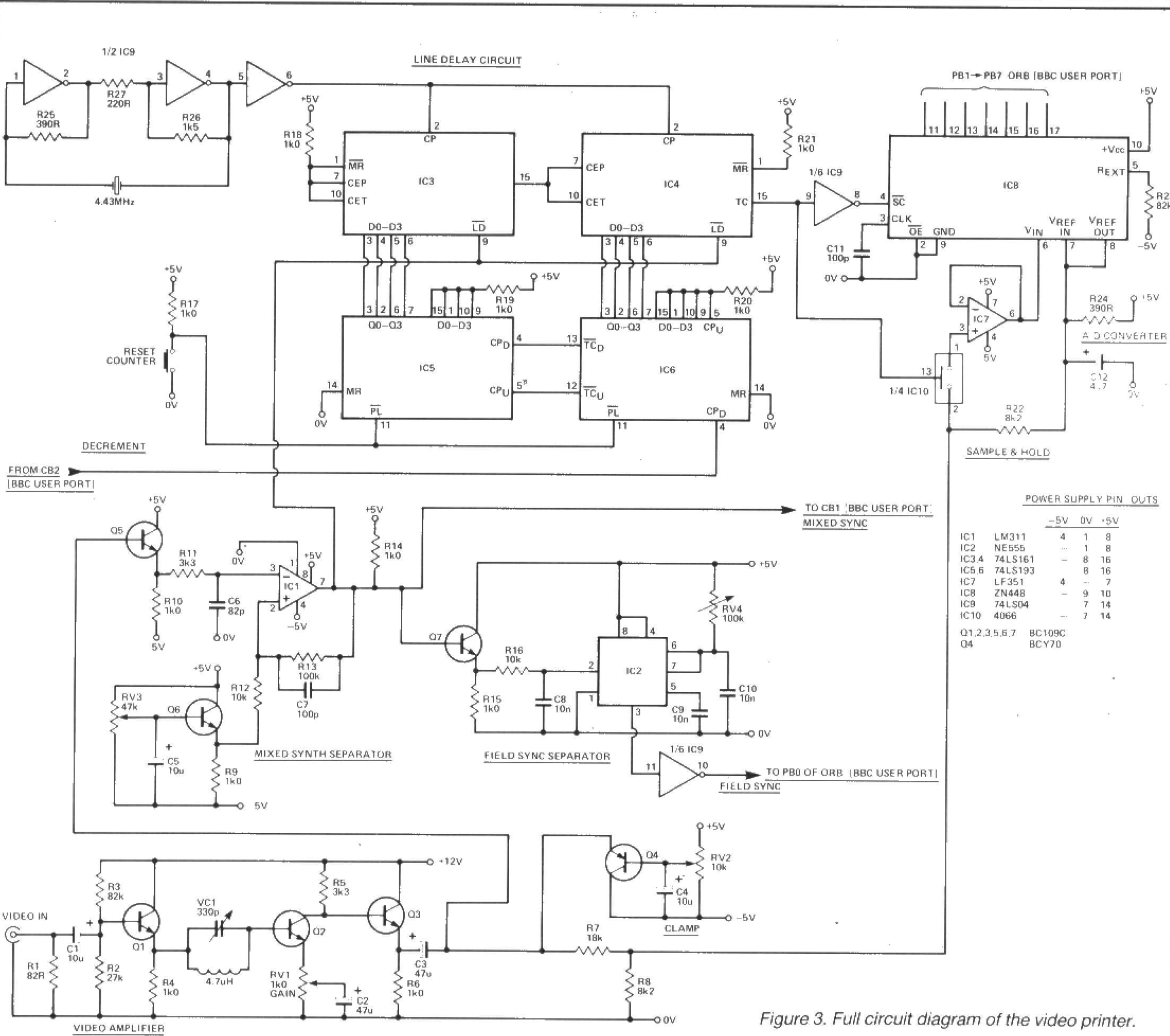


Figure 3. Full circuit diagram of the video printer.

range is effectively utilised since, when correctly adjusted, the video signal will range from 0V to –5V.

The Interrupt routine in the software reads and stores each latched byte from the A/D convertor into memory at a rate of once every line period. Once a complete vertical picture column has been captured the CB1 interrupts are disabled temporarily and CB2 decrements the delay parameter counters IC5 and IC6, ready for the capture of the next picture column. The frame sync is again detected and the picture column capture process starts again.

The process continues, with the delay byte being decremented (delay increased) for each consecutive picture column, until the whole picture is captured in memory. Note that it is possible to design the software so that data is selectively ignored, enabling lower pixel resolution or only 'windows' to be captured.

Parts List

Resistors

R1	82R
R2	27k
R3	82k
R4	1k
R5	3k3
R6	1k
R7	18k
R8	8k2
R9	1k
R10	1k
R11	3k3
R12	10k
R13	100k
R14	1k
R15	1k
R16	10k
R17	1k
R18	1k
R19	1k
R20	1k
R21	1k
R22	8k2
R23	82k
R24	390R
R25	390R
R26	1k5
R27	220R
RV1	1k
RV2	10k
RV3	47k
RV4	100k

Capacitors

C1	10μF
C2	47μF
C3	47μF
C4	10μF
C5	10μF
C6	82pF
C7	100pF
C8	10n
C9	10n
C10	10n
C11	100pF
C12	4μ7
VC1	330pF trimmer

Semiconductors

Q1,2,3,5,6,7	BC109C
Q4	BCY70
IC1	LM311
IC2	NE555
IC3,4	74LS161
IC5,6	74LS193
IC7	LF351
IC8	ZN448
IC9	74LS04
IC10	4066

Miscellaneous

4μ7H moulded inductor; push to make switch; IDC connectors for BBC User Port; 4.43MHz crystal.

Construction and setting up

The prototype was constructed on veroboard of size approximately 100 x 150mm. See the suggested layout of **Figure 4.** The wiring from the CMOS switch to the FET op-amp buffer should be as short as possible, otherwise layout is not critical. 100nF decoupling capacitors should be connected across the supply

rails every two TTL devices and a single 100uF capacitor is needed to decouple the video amplifier stage (this is not shown in the circuit diagram). Care should be taken when connecting up the +5V, –5V, +12,0V power lines, the expensive ZN448 does not tolerate anything above 5V!

A composite video colour bar source is required to set the circuit up, preferably from the same source that the digitiser is to be used with. Alternatively a suitable signal can be obtained from the BBC Micro itself using the simple program in **Listing 1.**

Set a 'scope to DC monitoring the output of the video amplifier between C3 and R7. The trace should be that of the standard 'staircase' type colour bar signal. If the signal source has colour, as well as luminance, the trace should appear 'fuzzy' because of the presence of the chroma subcarrier. V1C should be adjusted to reduce the chroma component to a minimum. Adjust the gain preset V1R so that the voltage difference between peak white and black level is just under 5 volts. Then adjust the clamp preset V2R so that the black level of the video signal sits on zero volts. The amplified signal is now compatible with the A/D Convertor.

Finally adjust V3R so that the mixed sync pulses appear at the output of IC1. V4R changes the width of the frame sync pulse which in turn determines the start row to be digitised. Check that the delay counter IC4 is generating carry pulses at pin 15. End of conversion pulses should be present at pin 1 of IC8, the A/D convertor, indicating that the device is indeed continuously converting.

LISTING 1.

```
 10 REM Program to generate
    colour bars
 20 REM of decreasing brightness
 30 MODE2:X1=0
 40 FORI=1TO8:READ C:GCOL0,C
 50   INC=1280/8
 60   MOVEX1,0:DRAWX1,1023
 70   PLOT85,X1+INC,1023
 80   PLOT85,X1+INC,0
 90   PLOT85,X1,0
100   X1=X1+INC
110   NEXT
120 DATA 7,3,6,2,5,1,4,0
```

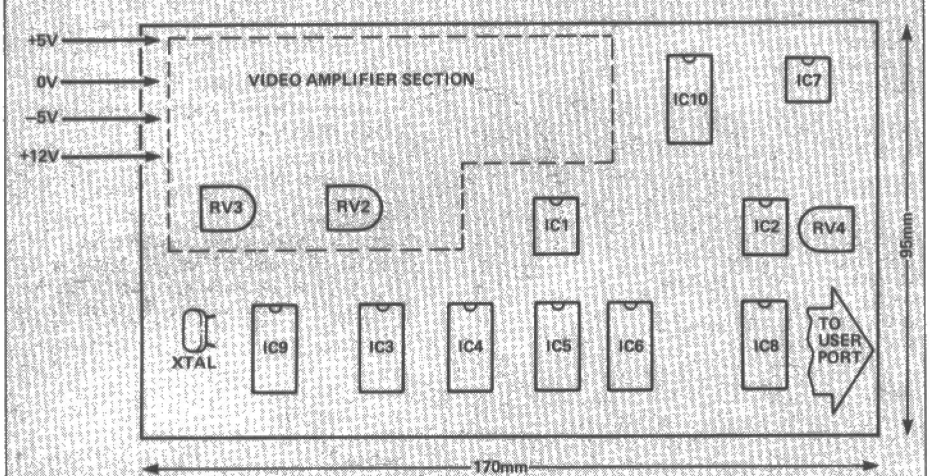


Figure 4. Suggested layout for the circuit. The prototype was consturcted on a 100 x 150mm piece of veroboard.

NEXT MONTH – Full listings of the software required to produce the picture prints.

THE CALCULATING QL

The QL's maths routines are built into QDOS rather than the SuperBASIC interpreter. Adam Denning shows how programmers may gain access to their power.

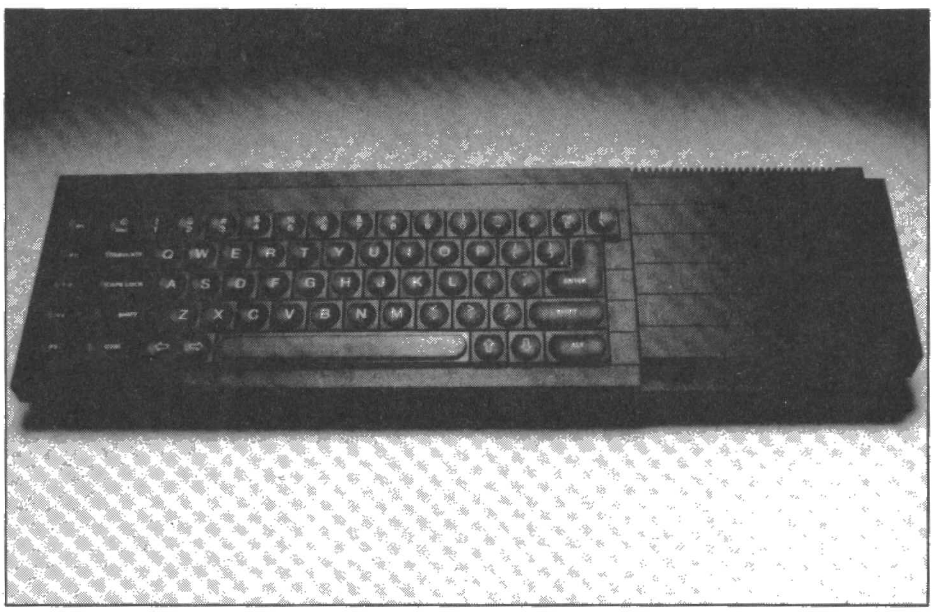
It is surprising just how many programs need to make complex calculations in order to achieve their aims. Games, for example, often need to work out the sines and cosines of numbers to move stars about on the screen.

Normally this involves the programmer in tedious and time consuming effort writing these routines, as there is no stock utility to do it all. The QL, however, is different. The mathematical routines are built into QDOS, the operating system, rather than the SuperBASIC interpreter. One of the reasons for this is that the QL's standard graphics routines are also part of the operating system, and use floating point format numbers for their arguments. The mathematical functions are all accessible from one routine called RI_EXEC.

This requires the operation code in D0.W and the address of a suitable arithmetic stack in A1. This stack address must be relative to register A6, but the beauty of 68000 machine code is such that this presents no problem. The arithmetic stack is a stack in the normal way, in that it grows downwards in memory, but each entry is normally a floating point number which occupies six bytes. Floating point numbers are formed so that the first two bytes form the exponent and the rest is the mantissa. The value represented by the number can then be found from this formula:

fp.value := mantissa * (2 ^ (exponent – $81F))

although we would very rarely need to apply this as RI_EXEC supports enough operations to perform most conversions for us. Some of the operations act on integers, which are two-byte signed numbers, and give a floating point result. Others take floating point numbers and convert them to integers or long integers. A long integer is a four-byte signed number. Unfortunately there is no routine to convert a long integer to a floating point number, but it's reasonably trivial to write.



Let's have a look at precisely what operations are available. **Table 1** shows them in numeric order, each code being the value that should be put in D0.W to perform that operation. TOS stands for 'top of stack' and NOS stands for 'next on stack'. The top of stack entry is of course the one at the lowest memory address.

Naturally all the trigonometrical operations expect their argument(s) to be in radians, but with the power provided by these operations it is very easy to write a 'degrees to radians' conversion routine. An extension of RI_EXEC, called RI_EXECB, allows a groups of operations to be performed in one go. When using this utility, A1 must still point to an arithmetic stack relative to A6, and A3 points to the list of operation codes, each stored as a byte value and terminated in a zero byte. Both routines preserve all registers except D0, which holds either zero or an error code on return, and the A1 stack pointer which is updated as appropriate. Some of the operations require D7 to be zero before use, but this appears to apply only to RI_EXP.

To demonstrate some simple uses for these routines, we'll write a couple of brief graphics programs to draw shapes using floating point co-ordinates. It is useful to know that the floating point representation of PI/2 is as PI/2 radians is 180 degrees, which is always a useful quantity to have in trigonometrical calculations. The last listing here is the code for a procedural extension to SuperBASIC which prints out the internal floating point representation in hex of any number which is passed as a parameter. The procedure's called TOFP,

TABLE 1.

Code (in hex)	Name	Operation
02	RI_NINT	round fp TOS to nearest integer
04	RI_INT	truncate fp TOS to integer
06	RI_NLINT	as NINT but to long integer
08	RI_FLOAT	convert integer TOS to fp
0A	RI_ADD	add TOS to NOS
0C	RI_SUB	subtract TOS from NOS
0E	RI_MULT	multiply TOS by NOS
10	RI_DIV	divide TOS into NOS
12	RI_ABS	absolute value of TOS
14	RI_NEG	negate TOS
16	RI_DUP	duplicate TOS
18	RI_COS	cosine of TOS
1A	RI_SIN	sine of TOS
1C	RI_TAN	tangent of TOS
1E	RI_COT	cotangent of TOS
20	RI_ASIN	arcsine of TOS
22	RI_ACOS	arcosine of TOS
24	RI_ATAN	arctangent of TOS
26	RI_ACOT	arcotangent of TOS
28	RI_SQRT	square root of TOS
2A	RI_LN	natural logarithm of TOS
2C	RI_LOG10	logarithm to base 10 of TOS
2E	RI_EXP	e to the power of TOS
30	RI_POWFP	NOC to the power of TOS

and it prints its data out to channel 1. We'll look at that a little later, but first the three simple graphics routines.

The program in **Listing 1** draws a box using the integer co-ordinates listed in the program. It starts off trivially by opening a console device using IO_OPEN and then it sets up a suitable arithmetic stack within its data area. As discovered in earlier issues, the register A5 points to the end of a job's data area upon invocation of that job, but A5 is relative to A6. This turns out to be ideal, as it is necessary for the arithmetic stack pointer to be relative to A6 too. Once the stack is set up a DBRA loop is entered starting at F_COORDS. This stacks an interger from the co-ordinates list onto the arithmetic stack, calls RI_EXEC with RI_FLOAT to convert this into a floating point number. This is done four times, controlled by another DBRA lkoop, so that the arithmetic stack now contains the (X,Y) co-ordinates of the start of the line and of its end. This information is just what the line drawing routine, SD_LINE, requires, but it needs its stack pointer to be absolute rather than A6 relative, so A6 is added into A1 before executing the trap. The outer DBRA loop ensures that four lines are drawn, and then the program stops doing anything meaningful by continually branching to the same instruction.

LISTING 1.

```
* A simple graphics demonstration program to draw a box
* By Adam Denning
          GET       "flp2_header_asm"
          SIZE      500

STACK     EQU       -50
          BRA.S     G_START            Ignore standard format code
          DC.L      0
          DC.W      $4AFB              Standard format ID word
          DC.W      8                  Job name
          DC.B      'GRAPHICS'
CO_ORDS   DC.W      10,10              X,Y of start of line
          DC.W      50,10              X,Y of end of line

          DC.W      50,10              X,Y of start of line
          DC.W      50,50              X,Y of end of line

          DC.W      50,50              X,Y of start of line
          DC.W      10,50              X,Y of end of line

          DC.W      10,50              X,Y of start of line
          DC.W      10,10              X,Y of end of line
G_START   MOVEQ     #IO_OPEN,D0
          MOVEQ     #-1,D1
          MOVEQ     #OPEN_NEW,D3
          LEA.L     DEVNAME,A0
          TRAP      #2
          LEA.L     CO_ORDS,A2         Set A2 to point to co-ordinates
          MOVE.L    #3,D2              Draw 4 lines
BOX_IT    LEA.L     STACK(A5),A1       Set A1 to suitable stack relative A6
          MOVEQ     #3,D1              Stack and convert 4 integers
F_COORDS  SUBQ.L    #2,A1              Make room for integer
          MOVE.W    (A2)+,0(A6,A1.L)   Stack co-ordinate
          MOVE.W    RI_EXEC,A3         Convert to floating point
          MOVEQ     #RI_FLOAT,D0
          JSR       (A3)
          DBRA      D1,F_COORDS
          ADDA.L    A6,A1              Make A1 stack absolute
          MOVEQ     #-1,D3             Infinite timeout
          MOVEQ     #SD_LINE,D0        Draw a line
          TRAP      #3
          DBRA      D2,BOX_IT          Next line
LOOP      BRA.S     LOOP
DEVNAME   DC.W      4
          DC.B      'CON_'
          END
```

Notice two things here. We have not used floating point numbers directly, but got the QL to do the conversion for us. Also, we could have made more efficient use of the co-ordinates in the list, as the end of one line is obviously the start of another.

The second program, shown in **Listing 2,** follows essentially the same principles, but draws a circle instead. It uses the SD_ELIPS trap routine, which needs its floating point parameters in the order:

LISTING 2.

```
* A simple graphics demonstration program to draw a circle
* By Adam Denning

          GET       "flp2_header_asm"
          SIZE      500

STACK     EQU       -50
          BRA.S     G_START            Ignore standard format code
          DC.L      0
          DC.W      $4AFB              Standard format ID word
          DC.W      7                  Job name
          DC.B      'CIRCLES',0

CO_ORDS   DC.W      50,50              X,Y of centre of circle

          DC.W      1                  Eccentricity of a circle is 1

          DC.W      30                 Radius of circle

          DC.W      0                  A circle has no angle of rotation
G_START   MOVEQ     #IO_OPEN,D0
          MOVEQ     #-1,D1
          MOVEQ     #OPEN_NEW,D3
          LEA.L     DEVNAME,A0
          TRAP      #2
          LEA.L     CO_ORDS,A2         Set A2 to point to co-ordinates
          LEA.L     STACK(A5),A1       Set A1 to suitable stack relative A6
          MOVEQ     #4,D1              Stack and convert 5 integers

F_COORDS  SUBQ.L    #2,A1              Make room for integer
          MOVE.W    (A2)+,0(A6,A1.L)   Stack co-ordinate
          MOVE.W    RI_EXEC,A3         Convert to floating point
          MOVEQ     #RI_FLOAT,D0
          JSR       (A3)
          DBRA      D1,F_COORDS

          ADDA.L    A6,A1              Make A1 stack absolute
          MOVEQ     #-1,D3             Infinite timeout
          MOVEQ     #SD_ELIPS,D0       Draw a circle
          TRAP      #3

LOOP      BRA.S     LOOP
DEVNAME   DC.W      4
          DC.B      'CON_'

          END
```

X,Y co-ordinates of circle's centre
Eccentricity of ellipse/circle
Radius of circle
Angle of rotation of major axis

The program (**Listing 3**) is almost identical to the box-drawing one, but this time there is little room for improvement as the circle's co-ordinates all refer to different things. Finally, we'll use SD_ARC to draw an arc for us.

LISTING 3.

```
* A simple graphics demonstration program to draw an arc
* By Adam Denning

          GET       "flp2_header_asm"
          SIZE      500

STACK     EQU       -50

* The floating point representation of PI/2 is $801 $6487 ED51
* We need this because PI/2 radians is 180 degrees

PI2_EXP   EQU       $801               Exponent
PI2_MAN   EQU       $6487ED51          Mantissa

          BRA.S     G_START            Ignore standard format code
          DC.L      0
          DC.W      $4AFB              Standard format ID word
          DC.W      4                  Job name
          DC.B      'ARCS'

CO_ORDS   DC.W      10,10              X,Y of start of arc

          DC.W      50,50              X,Y of end of line

G_START   MOVEQ     #IO_OPEN,D0
          MOVEQ     #-1,D1
          MOVEQ     #OPEN_NEW,D3
          LEA.L     DEVNAME,A0
          TRAP      #2

          LEA.L     CO_ORDS,A2         Set A2 to point to co-ordinates
          LEA.L     STACK(A5),A1       Set A1 to suitable stack relative A6
          MOVEQ     #3,D1              Stack and convert 4 integers

F_COORDS  SUBQ.L    #2,A1              Make room for integer
          MOVE.W    (A2)+,0(A6,A1.L)   Stack co-ordinate
          MOVE.W    RI_EXEC,A3         Convert to floating point
          MOVEQ     #RI_FLOAT,D0
          JSR       (A3)
          DBRA      D1,F_COORDS

          SUBA.L    #6,A1              Make room for PI/2 fp value
          MOVE.W    #PI2_EXP,0(A6,A1.L) Stack exponent
          MOVE.L    #PI2_MAN,2(A6,A1.L) Stack mantissa

          ADDA.L    A6,A1              Make A1 stack absolute
          MOVEQ     #-1,D3             Infinite timeout
          MOVEQ     #SD_ARC,D0         Draw an arc
          TRAP      #3

LOOP      BRA.S     LOOP

DEVNAME   DC.W      4
          DC.B      'CON_'

          END
```

The SD_ARC routine needs floating point parameters to specify the (X,Y) co-ordinates of either end of the arc, and then the angle which is subtended by the arc in radians. PI/2 radians is a reasonable angle, so after stacking the four co-ordinates and floating them, we manually stack the floating point representation of PI/2 and subsequently draw the arc.

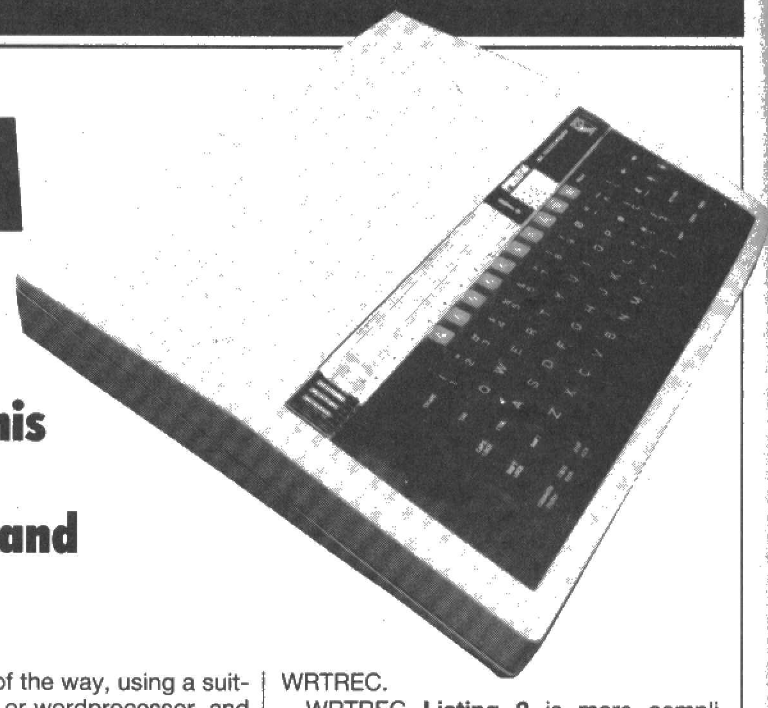
As there are always circumstances in which it is necessary to use a floating point number but can think of no way to easily obtain it, **Listing 4** adds that TOFP procedure to SuperBASIC.

It should be assembled and loaded into resident procedure space, and then its start address called. This installs it in the name list, but type NEW once first to make sure. It uses obvious techniques, taking advantage of CA_GTFP to gather its argument as a floating point number of SuperBASIC's arithmetic stack, and then it finds the channel ID of channel #1 using a few system variables. A conventional binary to ASCII hex routine converts the exponent and mantissa into strings which are subsequently printed to channel #1. This procedure is useful when we're playing around with RI_EXEC, as the human mind (well, mine anyway!) is not too good at converting a literal into internal format.

LISTING 4.

```
* A procedure to print out the hex internal representation of the floating
* point number passed as a parameter. The hex is printed to channel 1

          GET       "flp2_header_asm"

          MOVE.W    BP_INIT,A2
          LEA.L     PROC_DEF,A1
          JMP       (A2)

PROC_DEF  DC.W      1

          DC.W      TOFP_PROC-*
          DC.B      4
          DC.B      'TOFP',0

          DC.W      0,0,0

TOFP_PROC MOVE.W    CA_GTFP,A2         Get argument as floating point
          JSR       (A2)               !Note no argument checking!
          MOVEQ     #0,D1              Clear D1
          MOVE.W    0(A6,A1.L),D1      Get exponent in D1
          MOVE.L    2(A6,A1.L),D2      and mantissa in D2
          BSR.S     PT_HEX1            Print them
          MOVEQ     #0,D0              Clear D0
          RTS                          and return to Basic

PT_HEX1   MOVE.L    BV_CHBAS(A6),A2    Get base of Basic channels in A2
          ADDA.L    #CH_LENCH,A2       ..and channel 1 def. block
          MOVE.L    0(A6,A2.L),A0      Get channel ID in A0
          LEA.L     BUFFER+2,A1        Get ready to store string
          BSR.S     PT_HEX2            Convert exponent to hex chars
          MOVE.L    D2,D1              followed by mantissa
          BSR.S     PT_HEX2
          LEA.L     BUFFER,A1          Store string length...
          MOVE.W    #16,(A1)
          MOVE.W    UT_MTEXT,A2        ...and print it
          JSR       (A2)
          MOVEQ     #IO_SBYTE,D0       followed by a new line
          MOVEQ     #10,D1
          MOVEQ     #-1,D3
          TRAP      #3
          RTS

PT_HEX2   MOVEQ     #7,D0              Eight conversions per long word
H_LOOP    ROL.L     #4,D1
          MOVE.B    D1,D3
          ANDI.L    #$F,D3
          ADDI.B    #'0',D3            Convert to ASCII
          CMPI.B    #'9',D3
          BLE.S     H_NUMBER
          ADDI.B    #7,D3
H_NUMBER  MOVE.B    D3,(A1)+           Store character
          DBRA      D0,H_LOOP
          RTS

BUFFER    DS.B      18

          END
```

RANDOM ACCESS



Adam Denning concludes his series by giving some subroutines for the writing and erasing of records.

Last month we wrote the routines to initialise and terminate the random access filing system, and the routine to read a specific record. Now we need to design the subroutines for writing and deleting records.

The routine to write a record is called WRTREC, and is called with the number of the record which is to be written to disk in RECNUM. Four bytes in OSBLOK must be initialised with the address from which the record is to be written, just like RDRECD. If the specified record already exists, its present contents is overwritten without further ado. If the record does not exist, then one of two actions is taken:

1) The record map is searched for the first invalid (ie deleted) record, which is signified by a record number of $FFFF. This record is overwritten by the new record, and the $FFFF is replaced with the real record number.

2) If there are no invalid records then the new record is appended to the file and the map updated accordingly.

The process of deletion is fairly straightforward; the record number in the record map is simply overwritten with $FFF. No other action needs to be taken. If the specified record cannot be found, an error is returned.

Searching

Let's examine the code for DELROD first **(Listing 1)**, as this is simple. It is entered with the record number in RECNUM.

This routine calls a subroutine with the name of 'SEARCH'. We do not need to write this, as it already exists in RDRECD – it's the section of code from the start of RDRECD to BADREC. Both it and RDRECD need a slight bit of alteration to allow the conversion, but it's very simple: change the start of RDRECD to look like this:

```
RDRECD  JSR   SEARCH
        BNE   GOTREC
        RTS
GOTREC  .....
```

Move the code which forms the subroutine down out of the way, using a suitable screen editor or wordprocessor, and alter the line which says 'BEQ GOTREC' to 'BEQ FOUND'. Then insert a couple of lines after BADREC:

```
FOUND   LDA   #$FF
        RTS
```

Now back to DELRCD: If the record cannot be found, the routine returns with the zero flag set. If the record is found, $FF is loaded into the relevant locations in the record map, using (CURECD),Y addressing. Y is then decremented to reset the zero flag and the routine returns. We do not decrement the number of records in the file, as the record still notionally exists and may be discovered in subsequent calls to WRTREC.

LISTING 1.

```
* Random Access filing system using a record map
* By Adam Denning 10th December 1984
* (C) 1984 Adam Denning
* The second part - this must be APPENDED to the source in part 1 either by
* using ADE's CHN directive or by physically joining the files together
* And the 'search' and 'setptr' parts of RDRECD must be converted to subroutines


* The 'delete a record' routine
* Entered with desired record number in RECNUM
* Returns with the zero flag set if the record could not be found,
* reset otherwise.


DELRCD  JSR     SEARCH
        BEQ     DELEND          ; returns with zero flag set if no such record

DELREC  LDA     #$FF
        STA     (CURECD),Y
        DEY
        STA     (CURECD),Y
        DEY                     ; return with zero flag reset - success!
DELEND  RTS
```

WRTREC **Listing 2** is more complicated than RDRECD. First call SEARCH to see if the record number given already exists, and if it does branch straightaway to RWRITE. Otherwise, preserve the given record number in SPARE, put FFFF in RECNUM, and call SEARCH again. This will find the first invalid record entry and leave its map address in CURECD. This is set to the true record number, then return the record number to RECNUM once more.

Then we branch to RWRITE. If there are no invalid records append the new record to the end of the map. This is done by taking the address of the map plus two, to a point past the record count, and then by

LISTING 2.

```
SPARE   EQU   $77

WRTREC  JSR   SEARCH
        BNE   RWRITE
        LDX   RECNUM
        LDY   RECNUM+1
        STX   SPARE
        STY   SPARE+1
        LDA   #$FF
        STA   RECNUM
        STA   RECNUM+1
        JSR   SEARCH
        PHP
        LDX   SPARE
        LDY   SPARE+1
        STX   RECNUM
        STY   RECNUM+1
        PLP
        BEQ   NEWRCD
        TYA
        LDY   #1
        STA   (CURECD),Y
        TXA
        DEY
        STA   (CURECD),Y
        DEY
        BNE   RWRITE

NEWRCD  LDX   #>MAPLOC+2
        LDY   #<MAPLOC+2
        STX   SPARE
        STY   SPARE+1
        CLC
        LDA   MAPLOC
        ADC   SPARE
        STA   SPARE
        LDA   MAPLOC+1
        ADC   SPARE+1
        STA   SPARE+1
        LDA   MAPLOC
        ADC   SPARE
        STA   SPARE
        LDA   MAPLOC+1
        ADC   SPARE+1
        STA   SPARE+1
        LDY   #0
        LDA   RECNUM
        STA   (SPARE),Y
        INY
        LDA   RECNUM+1
        STA   (SPARE),Y
        INC   MAPLOC
        BNE   RWRITE
        INC   MAPLOC+1
RWRITE  JSR   SETPTR
        LDA   #1
        LDX   #>OSBLOK
        LDY   #<OSBLOK
        JSR   OSGBPB
        RTS
```

adding the number of records in the file on twice to get the address of the end of the map. The record is then stored at this address and the count of the number of records in the map (and therefore in the file) is incremented. Then we get to RWRITE.

This part of the code is virtually identical to the corresponding section in RDRECD, except that we put 1 rather than 3 in the accumulator before calling OSSBPB. This command makes the routine write the specified number of bytes from the specified base address to the file, using the sequential pointer value in which OSBLOK is placed.

We have converted another part of RDRECD into a subroutine **Listing 3** – the section which converts the record number into a sequential file pointer by multiplying it by the length of a record. Follow the same technique as in converting SEARCH, but all that needs to be added is an RTS just before the LDA #3, and a call to SETPTR at GOTREC. SETPTR is then the section of code from the old GOTREC instruction to the newly added RTS. The final result is shown in **Listing 3.**

LISTING 3.

```
RDRECD  JSR   SEARCH
        BNE   GOTREC
        RTS
GOTREC  JSR   SETPTR
        LDA   #3
        LDX   #>OSBLOK
        LDY   #<OSBLOK
        JSR   OSGBPB
        LDA   #$FF
        RTS
```

The only section to be written now is the program which needs to use a random access filing system. It's been a fairly long and unavoidably intermittent journey, due in the most to the greater joy of experimenting with a new computer (the QL of course!), but we should all be in a position now to use the random access capabilities of the BBC Micro with a disk interface to their full potential.

DAVID KARLIN

Why Q COM is everything you could wish for in communications.

The QL is now communicating – via Q COM! This exciting three-part peripheral presents QL users with a multitude of ways to exploit the world of communications.

Once connected to the QL, Q COM allows you to access the considerable number of phone-in databases, such as Prestel and QNet.

Q COM enables you to communicate with other computer users. Its facilities include electronic mail, data transfer from Microdrives and bulletin boards.

Through it you can link your QL to larger minicomputers. Q COM has full capability in this area, and allows the QL to talk to powerful mainframes.

Q COM's automatic dialling and call acceptance facilities, together with the storage of messages from other modems, will revolutionise the way you use your telephone.

The next few pages of QL News tell you much more.

It's enough for me to say here that with the QL and Q COM, you'll be exploring new openings in communications for some time to come!

David Karlin

David Karlin,
Chief Design Engineer.

The Q COM package

Three special parts to stack!

Q CON

QL communications interface

This multi-speed interface contains the sophisticated software used to set up QL communications – and to control the Q CALL and Q MOD units.

Q CON also comes complete with Microdrive-based software. This enables the QL to link to larger computers using VT100 and viewdata protocols.

The software will also run *any* standard modem – connected via Q CON's built-in RS-232-C port.

Most importantly, Q CON allows the QL to transmit and receive at rates switchable from 75 to 9600 baud (encompassing the widely-used 75/1200 Prestel rates, and 1200/1200 half duplex rates for user-to-user exchange).

Q CON is specially styled to suit your QL – with similar fluting and ribs – and forms the base module of a vertical-stacking system.

It's supplied with full instructions, software on Microdrive cartridge, and connecting leads.

Q CALL

QL auto dial/answer unit

Q CALL gives every QL user something out of the ordinary.

It's a module which links directly to your telephone, and allows auto-dialling at the push of a single key. In the same way, it will permit incoming calls to be accepted automatically . . . and even trigger pre-programmed activity from the QL!

Q CALL is the central unit of the package. It plugs directly into Q CON – so there are no connecting cables to worry about.

Q MOD

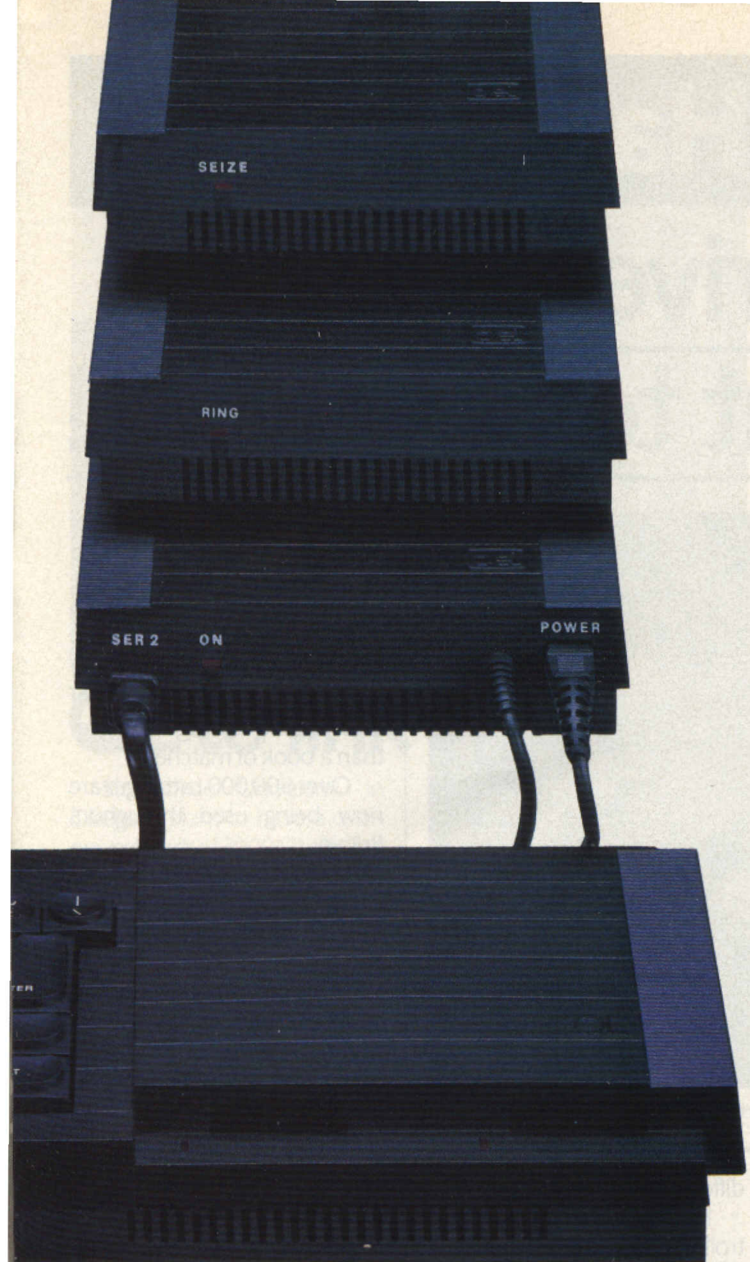
QL modem

Q MOD has all the powerful facilities expected of a modem, in a neat and simple unit.

It uses either V23 75/1200 or 1200/1200 baud rates, for Prestel, Micronet 800 and all the viewdata services described alongside.

It also incorporates a telephone extension socket for manual dialling.

Q MOD is the top unit of



Q COM, and comes with a 9' built-in telephone cable.

All three units are available from Sinclair on (0276) 685311 and from selected Sinclair stockists.

The QL hooked on voice and data

The QL can now act as your personal address book and telephone operator!

Q COM allows you to store hundreds of personal or business numbers.

You can store lengthy passwords and account numbers – and recall them – at the touch of a single key.

And any information that's sent to you from other modem-owners can be gathered and stored on Microdrive cartridge, or incorporated into your QL Quill documents!

Exploring the world of QNet, Prestel, Micronet and more!

Thousands of QL users already enjoy the excitement of linking to a nationwide mainframe.

Q COM turns your QL into an intelligent terminal, allowing you to access many thousands of pages of information, software and communications facilities.

The services brought to you through Prestel can include Micronet 800, Viewfax 258 and QNet, the new QL database.

Membership of QNet will bring you free software, QL news and features, and all the wide-ranging services of viewdata!

If armchair shopping is more your style, that's easy too. It's often possible to place a direct order using your QL! For dedicated QL owners, there's a daily selection of software reviews, chart toppers . . . and all the facts and figures you need to make buying peripherals simple.

With Q COM you can also 'download' software from the system directly into your QL and either use it immediately, or store it on Microdrive cartridge.

In fact the only problem you'll face with a viewdata service is finding enough time to explore its many features!

You can find out how to join QNet by phoning 01-278 3143.

News . . . information . . . banking services and QNet. And only a fraction of the QL's new viewdata capability.

QL meets the mainframes!

The Q CON unit of Q COM turns your QL into a VT100 terminal, providing instant access to in-house computing services, both mainframe and mini.

Whether you are using your QL at home or at work, Q COM gives you access to electronic bulletin boards which provide help and advice 24 hours a day. You can leave messages or notices for friends or business contacts and even hold live discussions with them.

Additional benefits for the QL business user include easy access of in-house company software, and the interrogation of other data bases around the country.

There's also the opportunity of linking to British Telecom Gold – the widely-publicised and popular messaging service.

THE DIAGRAM DIAGNOSED

Mike James makes a careful study of the BBC micro's circuit diagram (printed on our centre pages) and finds it still one of the most innovative designs around.

The BBC Micro is still one of the most innovative microcomputer designs from both the software and the hardware point of view. The internal design of the BBC's software has been the subject of a number of articles in *E&CM* and it is the purpose of this article to redress the balance in favour of the hardware.

When contemplating the BBC Micro's hardware it is all too easy to go on at length about its 'larger' features such as the Tube or its ability to page ROMs but the machine is also interesting at a more detailed level. If you examine its circuit diagram it is not long before you spot some additional clever aspects of the design. Sometimes it is simply the way a timing signal has been derived, or something to do with the way the different parts of the hardware work together. Reading circuit diagrams is not often a rewarding experience but in the case of the BBC Micro it certainly is! To help get you started the following notes provide a general overview of each section of the circuit and highlight a few of the special features. Throughout it is assumed that you are a BBC Micro user, know something about computer design, have a TTL data book and have or can acquire data sheets on the other chips in the design.

The BBC Micro uses a wide range of clock pulses to synchronise its operation. Although the 6502 CPU (IC1) is a double speed device running at 2MHz, the system's dynamic RAM is accessed four times in every microsecond; twice by the 6502 and twice by the 6845 CRTC (Cathode Ray Tube Controller). Thus the RAMs are cycled at 4MHz. A 16MHz crystal oscillator (IC43 and IC40) provides the master clock that is divided down by the video ULA (IC6) to 8, 4 and 1MHz (all in phase). A 2MHz clock for the 6502 is provided by OC30 by phase shifting the original 2MHz clock from IC6. As some of the peripherals that the 6502 has to access to are single speed devices IC33, IC31 and IC34 are used to stretch the clock to 1MHz when an appropriate address is detected by IC23. A 6MHz clock is generated by IC40, IC38 and IC37 for use by the teletext character generator IC5.

A power on reset signal is generated by a 555 timer (IC16) and a simple RC network (R20 and C10). The reset signal from the RC network is only applied to the system 6522 VIA (IC3). The 555 can also be triggered by a reset switch (which is not normally fitted to the main PCB) or by the BREAK key on the keyboard. Thus, by examining the state of the system VIA, the machine can discover if a reset is due to power on. Obviously following power on there is no valid data in memory but after the reset button is pressed there may well be.

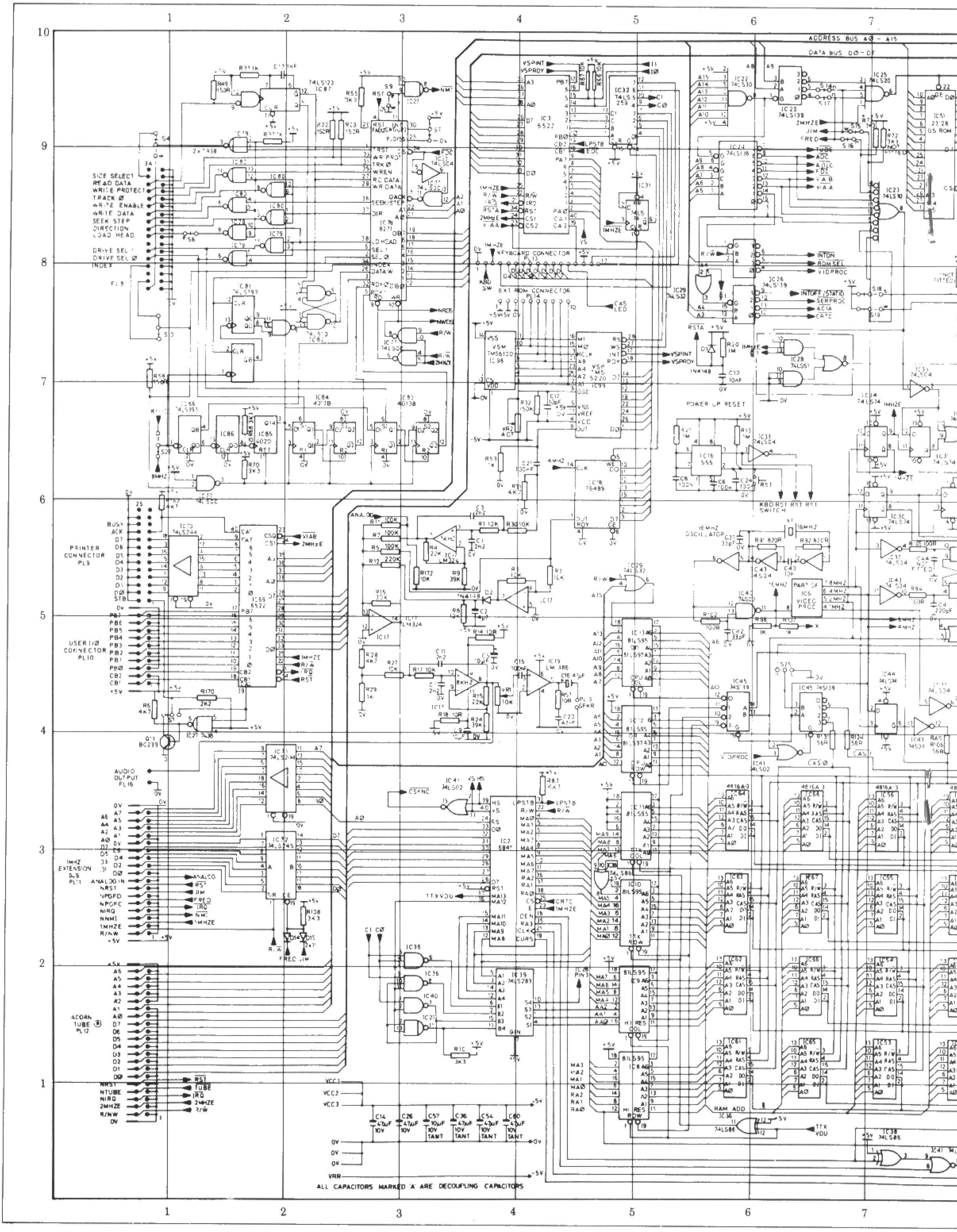
The CPU's address bus is distributed around the rest of the circuit but individual chip select signals are derived by IC20, IC21, IC22, IC24, IC25, IC26 and IC29. Chip selects for the Tube, A to D, Econet interface, floppy disc controller and the two VIAs are provided by IC24 (which uses IC22 as a high address detector). Two of the outputs of IC24 are further decoded by IC26 to provide further chip selects for the Econet station number register, the ROM select latch, the video ULA, the serial ULA, the 6850 ACIA and the CRTC. Notice that the R/W line is used in the decoding of these first four select lines which means that it is only possible to write to the ROM select latch and to read the station ID(entifier) – that is they both occupy the same address – and it is only possible to write to the video ULA. The address decoding for the ROMs is straightforward apart from the use of IC76 to select which of the four paged ROMs is used. The expansion I/O areas Fred and Jim are selected by outputs from IC20. The RAM is selected by simply using address line 15 to indicate accesses in the lower 32K. A signal for slow (1MHz) devices is derived by IC23.

The BBC Micro's 32K of RAM is provided by sixteen 4816 dynamic RAM chips – IC53 to IC68. As already mentioned these are accessed four times every microsecond and this implies that they are very fast devices. The row and column addresses are supplied by six octal buffers IC8 to IC13. The CPU accesses the RAM via IC12 and IC13 and the CRTC via IC8 to IC11. Which device has access is controlled by IC45. The dynamic RAM is refreshed by the CRTC continually cycling through all of the possible row addresses. The data bus to the CPU and other devices is buffered by IC14.

Perhaps the cleverest part of the entire BBC Micro's circuit is the way that the 6845 CRTC is used. If you look at an application sheet for the 6845 you will see that it is supposed to address two different types of memory. An area of RAM, used to store ASCII codes and an area of ROM to generate the dot patterns of the characters to be displayed. The idea is that the output of the RAM is used to select a dot pattern from the ROM and the CRTC provides extra addressing to select each byte of data which make up the characters. Of course the BBC Micro (in modes 0 to 6) doesn't use a ROM to generate the dot patterns as they are stored directly in RAM. Another interesting feature is the way that the 6845 is coaxed into providing colour information – the raw 6845 is a monochrome device. The key to understanding how the 6845 has been used in the BBC Micro is to notice that although its address lines are segregated into two groups – MA0 to MA13 and RA0 to RA2 – they can just as easily be used to address a single block of memory. In other words, the CRTC simply addresses the memory a certain number of times per TV scan line and what happens to the data that is produced is not its concern.

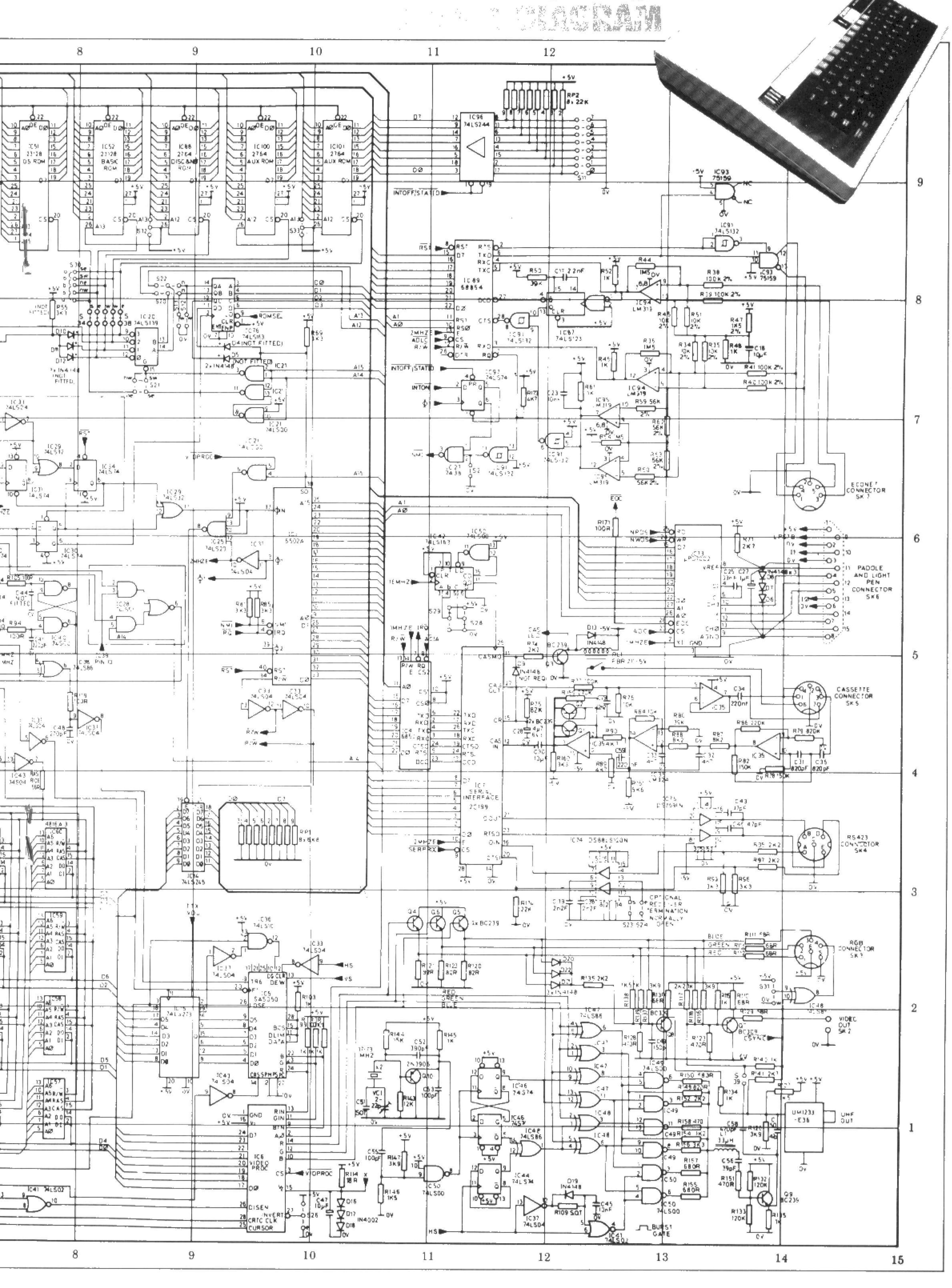
Once the data has been selected by the CRTC it is sent (a byte at a time) to the video ULA (IC6) where it is serialised and applied as the address to a small area of static RAM – the palette RAM. This is responsible for adding the colour to the display. The output of the palette RAM is a four bit word indicating the state of the R(ed), G(reen), B(lue) and Flash signals. The RGB signal is fed (via Q4 to Q6) to the RGB monitor connector and to the PAL colour encoder formed by Q10, IC46, IC47, IC48, Q8 and Q9. The PAL signal is finally

COMPUTING



ADDRESS BUS A0 – A15
DATA BUS D0 – D7
PRINTER CONNECTOR PL9
USER I/O CONNECTOR PL10
AUDIO OUTPUT PL16
1MHZ EXTENSION BUS PL11
ACORN TUBE PL12
KEYBOARD CONNECTOR PL13
EXT ROM CONNECTOR PL14
POWER UP RESET
16MHZ OSCILLATOR
KBD RST SWITCH
RAM ADD
ALL CAPACITORS MARKED 'A' ARE DECOUPLING CAPACITORS

BBC MICRO



IC51 23128 DFS ROM
IC52 23128 BASIC ROM
IC88 2764 DISC&NFS ROM
IC100 2764 AUX ROM
IC101 2764 AUX ROM
IC96 74LS244
RP2 8x 22K
IC93 75159
IC89 68B54
IC94 LM319
IC87 74LS123
IC91 74LS132
IC92 74LS74
IC20 74LS139
IC76 74LS163
IC21 74LS00
IC25 74LS27
IC29 74LS32
IC30 74LS74
IC31 74LS04
IC34 74LS74
6502A
IC42 74LS163
IC52 74LS00
IC73 UPD7002
IC4 6850
IC7 SERIAL INTERFACE 2C199
IC74 74LS245
RP1 8x 6K8
IC35
IC75 DS88LS120N
IC74 DS88LS120N
IC36 74LS10
IC33 74LS04
IC15 74LS273
IC5 6845
IC6 VIDEO PROC
IC43 74LS04
IC41 74LS02
IC46 74LS74
IC47 74LS86
IC48 74LS86
IC49 74LS00
IC50 74LS00
IC44 74LS74
IC37 74LS04
IC42 74LS02
UM1233 -E36
ECONET CONNECTOR SK7
PADDLE AND LIGHT PEN CONNECTOR SK6
CASSETTE CONNECTOR SK5
RS423 CONNECTOR SK4
RGB CONNECTOR SK3
VIDEO OUT SK2
UHF OUT
BURST GATE
17.73 MHZ
3x BC239
BLUE
GREEN
RED
INTOFF/STATID
INTON
ROMSEL
VIDPROC
2MHZE
1MHZE
HS
VS
8
9
10
11
12
13
14
15
1
2
3
4
5
6
7

applied to a standard ASTEC UHF modulator and brought out through a phono socket. A composite monochrome video signal is also provided by Q7 and Q8 for monochrome monitors. Notice that all the TV sync signals are generated by the CRTC. The cursor signal is mixed into the video signal by the video ULA.

Teletext

There is one graphics mode where the CRTC is almost used as its makers intended – Mode 7. In this mode the CRTC addresses the top 1K of RAM to obtain ASCII codes that are applied to a teletext character generator – IC5. Even here however there are some tricky methods used. CRTC address line MA13 is used as a teletext indicator and this is used to select between two pairs of octal buffers that apply the rest of the CRTC's address lines to the RAM. Modes 0 to 6 use octal buffers IC8 and IC9 and Mode 7 use IC10 and IC11. The reason for this is that the CRTC's row address lines are not applied to the teletext ROM and so the same ASCII data has to be presented to it more than once. This implies using a different addressing scheme for the RAM and this is accomplished by using two sets of buffers. The RGB output of the Teletext ROM is applied to three input lines of the video ULA and hence on to the rest of the video circuitry.

Hardware scrolling

The time it takes to scroll a screen increases with the amount of memory involved. As the BBC Micro uses up to 20K of memory for a display, simple software scrolling would take rather a long time. To speed things up a little a very ingenious system of hardware scrolling is used. The CRTC contains a register that holds the address of the first character that will be displayed. It is obvious that a scroll can be achieved simply by incrementing the value in this register by the number of memory locations used to store a line of text. The only trouble is that the area of memory addressed will go beyond the 32K RAM limits. Addresses that are above the 32K limit should obviously be reduced to refer to the start of the display area. This can be done by adding a constant to the address that the CRTC generates. The constant is generated by IC37, IC36 and IC40 from the two lines C0 and C1 that indicate the currently selected display mode. The constant is added to four of the high order address lines generated by the CRTC by IC39.

The system VIA

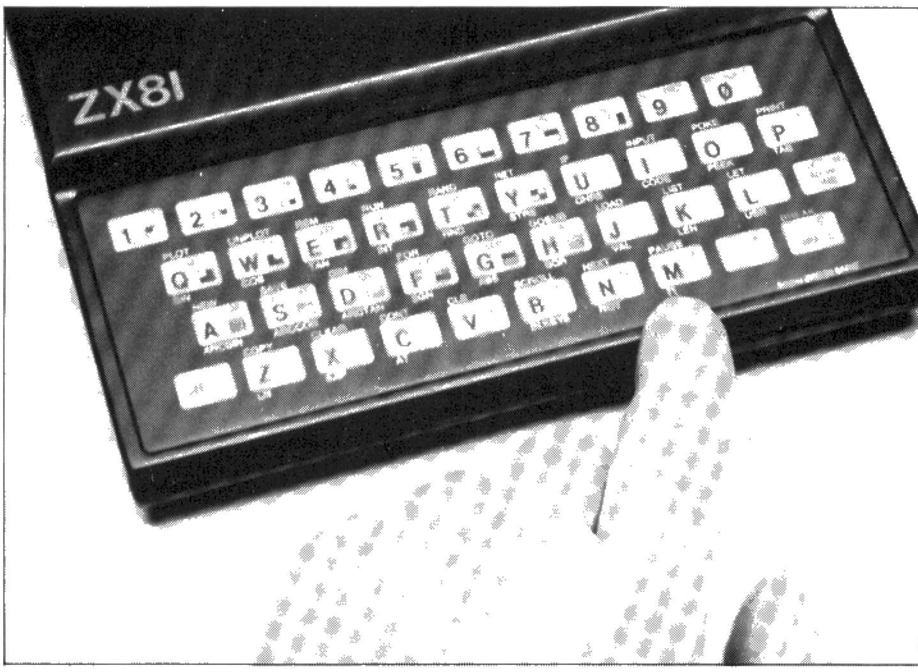
The CA1 input is used to interrupt the processor if any key on the keyboard is pressed, the CA2 input is used to signal the start of a new TV frame, the CB1 input signals the end of an A to D conversion and finally the CB2 input is used to interrupt the processor in response to a pulse from a lightpen.

The rest of the VIA's I/O lines are best thought of as forming a sort of auxiliary peripheral data and address bus. The eight A side data lines are used as a bi-directional data bus between the processor and a number of other chips (including the keyboard and sound generator). The first four B side data lines are used as an 'address bus' to select or activate a number of devices. This is achieved by use of an addressable latch (IC32). The first three B side data bits, PB0 to PB2, are used to address a location in the latch and PB3 is used to set the state of the location. (The latch is clocked by IC31). In this way four VIA lines can be used to set the state of the eight outputs of the latch! Bit zero of the latch enables the sound chip (IC18), bits one and two control the speech chip (when fitted), bit three enables the keyboard, bits four and five provide the signals C0 and C1 used to indicate the mode during hardware scrolling and bits six and seven control the caps lock and shift lock LEDs on the keyboard. The remaining B side data lines are configured as inputs. PB4 and PB5 are used to detect the state of the 'fire buttons' on the joystick interface and PB6 and PB7 are status inputs from the speech synthesiser.

CONTINUED NEXT MONTH

A question of scale

John Mellor explains why VLSI (that's Very Large Scale Integration) is an integral part of the microcomputer revolution.



The home computer user probably first became aware of custom integrated design when Clive Sinclair produced the ZX81 four years ago. By the use of a single customised circuit he was able to produce an upgraded version of the ZX80 which consumed less power and cost less to manufacture. A Ferranti Uncommitted Logic Array (ULA) was employed to replace 14 standard TTL components.

Standard components are those such as the TTL range that are available to all of us from electronic components suppliers. The use of standard components means that anyone can put together a Z80 or even an Apple computer with a little know-how. The Japanese got into trouble for doing just that but not before they had made a lot of money without incurring any development cost. So, not only are you able to make the computer more financially realistic by using custom circuits, but you also stop the opposition ripping-off your design.

VLSI stands for Very Large Scale Integration. We all have access to VLSI technology now and many of you will have used it. The most commonly available VLSI chips are large dynamic RAM chips and microprocessors. These are standard components which contain more than 10,000 gates.

The large number of gates is achieved by making the area of the silicon chip larger, reducing the size of the transistors and the width of the interconnections, and redesigning the logic functions so that they use fewer transistors. In other words, you can double the number of gates per chip by either doubling the chip area, halving the geometry or halving the number of transistors used to provide the functions.

A combination of advances in these areas has resulted in a doubling of the number of gates per chip every year. The result is a wide range of more powerful chips such as video and disk controllers, maths processors and interface devices. This makes the job of the systems designer a lot easier – imagine trying to build a disk controller from TTL NAND gates! – they had to before the LSI chips became available.

The major impact of VLSI on digital systems is the capability it gives to design custom and semi custom chips. A section of logic consisting of 32 x 3 input AND gates, 8 x 4 input NOR gates and 12 inverters was recently replaced by a single 20 pin PAL (Programmable Array Logic). PALs are readily available for £7 and can be programmed at home using a modified ROM programmer. The original circuit used 11 x 7411, 4 x 7425 and 2 x 7406: a total of 17 TTL chips plus a fair area of PCB. Even with less complex circuits it is economical to customise your own piece of silicon. For £7 a great deal of logic design and PCB layout work can be saved.

Is CAD necessary?

A common misapprehension is that you cannot design a semi-custom chip without a lot of expensive CAD software and a mainframe computer with graphics workstation attached. This software is needed only for a very large circuit design which will require a change of approach by the designer implementing a complete system on silicon.

To design the Sinclair ZX81 ULA the Ferranti logic designers took a large drawing of the ULA showing all the cells, and then drew the connections to the various cells required to produce the logic – they used coloured pencils. Each cell consisted of four transistors and a dual current source. In addition each cell had three resistors (the newer ULAs have six transistors and a quadruple current source).

The transistors in a cell can be connected to form inverters or basic 2 x 2 input NOR gates. Three cells were connected to form a D-type flip-flop. No pre-layout logic simulator was required because the prototype logic existed in the Sinclair Z80 – only a little thought about where to pencil in the connections was needed.

The Ferranti ULAs are to my knowledge the only bipolar arrays available. Most other manufacturers use CMOS technology. The silicon wafers are pre-fabricated and have gone through all but the final stage of production. The surface of the chip is covered in a layer of aluminium.

". . . VLSI – kingpin of the microcomputer revolution . . ."

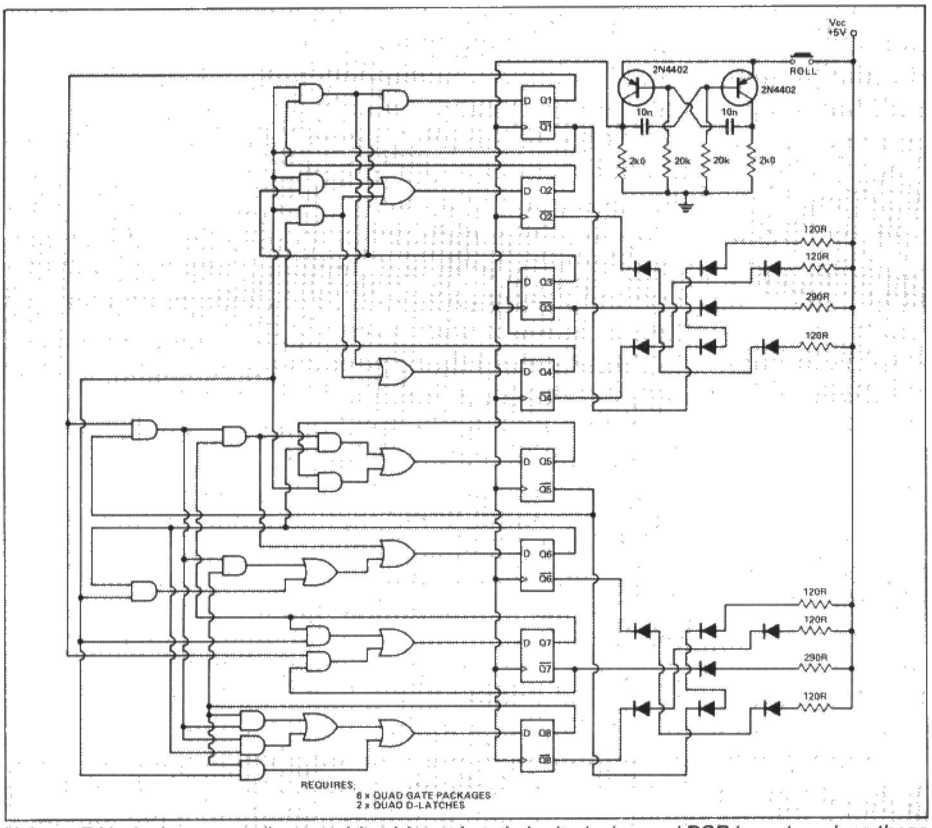
It is this layer which will be etched away to leave the required interconnections. The actual process of committing the chip is similar to etching a PCB but on a smaller scale.

This cannot be done at home because the equipment to produce the mask which contains the etching pattern, the chip testr, passivation, lead bonding and packaging is complex and expensive.

The interconnections have to be laid out on a high resolution computer terminal or digitised by using an accurate plotter to enter the route of each interconnect into a

computer. The computer then checks the layout for any shorts, open circuits and other design rule faults. It uses a test program, which can be a truth table, to test the logic functions.

Acorn computers recently set up a division purely to research into custom and semi-custom chip design. They know that to remain competitive in the home and business micro market they need to have in-house chip design expertise. They were the first to perform design in house, most other manufacturers send a logic diagram, DC specification and factual description or truth table to the semi-custom manufacturer.



Using a PAL device can realise considerable savings in logic design and PCB layout work as these two functionally equivalent circuits show.

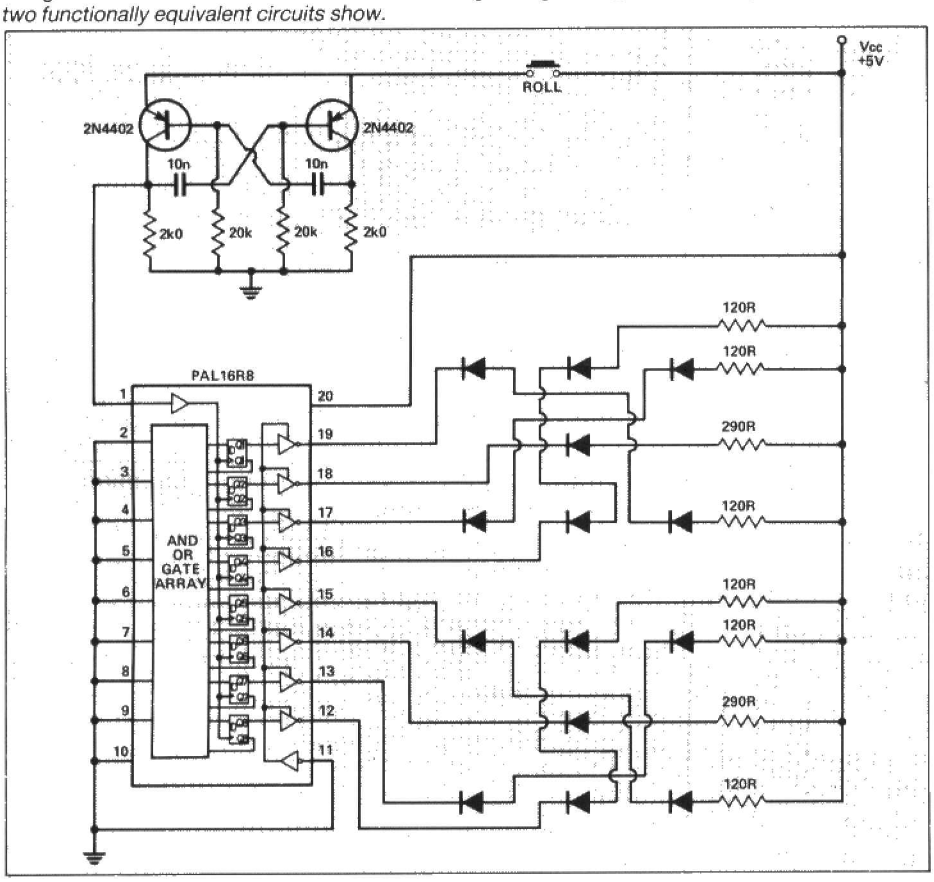


TABLE 1.

Year	Computer	Number of Custom ICs used
Pre 1981	ZX80, PET, Apple, Tandy, Acorn, Tangerine, Nascom	0
1981	ZX81	1 (F)
1982	VIC	1 (VIC CHIP-C)
	Spectrum	1 (F)
	BBC	1 (F)
	ORIC	1 (CD1)
	Aquarius	1 (F)
1983	Interface 1 + MDV	2 (F)
	Interface 2	1 (F)
	Commodore 64	1 (C)
1984	Electron	1 (A)
	Amstrad	1 (F)
	Advance	9 (F)
	QL	4 (2F + 2S)
	Enterprise	2

F = Ferranti, C = Commodore, A = Acorn, CD1 = California Devices, S = Synertic (Plessey, NCR).

The Enterprise, the QL and the Electron all make extensive use of custom integrated circuits. Circuits which are specific to that one application in that one product. By taking this route the manufacturers hope to remain competitive, in the first instance at home, against foreign and home competition.

Even Sinclair, who previously only employed a small team to conceptualise a design, and therefore relied heavily upon outside help for the realisation and manufacture of the hardware, is setting up his own IC design and fabrication facilities. But custom design can lead to problems. The home micro market is a good example of a situation where it is important to make the product available before the opposition does. A fault in the design of a PCB based logic system consisting of standard components can often be corrected by the addition of links and the breaking of tracks. Many early modification runs contain at least one modification of this type before the circuit is corrected. But if an error has been made in a semi-custom circuit only the fortunate can correct it by an external modification (as Sinclair did with the Spectrum).

Computer designers are not yet experienced enough at specifying logic circuits to cover all functional variations, thus the chip produced by the chip manufacturers may work to the specifications supplied but could still contain bugs. The Same of course is true of programs.

The redesign, simulation, fabrication and testing of a chip can take months. If a second bug is then discovered the delay and unexpected additional expense can be disastrous and the company may never recover from the setback. The late arrival of the Enterprise is due in part to bugs that were discovered in the custom ICs.

DRAGON JOTTINGS

Mike James shows how the Dragon's graphics facilities can be expanded to rival those of the BBC micro.

The Dragon sports a reasonably good low resolution and an excellent high resolution graphics capability. However one common and very useful feature that it lacks is the ability to define new graphics characters. To be more precise, it has a clear division between text and graphics that means that it is impossible to PRINT any characters on a hi-res screen – even the standard pre-defined characters. This makes the user-defined character problem even more important because if you can find a way of printing any graphics shape on a hi-res screen you can also print all of the standard text characters.

Although the most obvious method of producing small shapes in this way is to use the DRAW command this article approaches the problem by using GET and PUT. A few simple ideas combined with the standard PUT command gives the Dragon user-defined graphics that are as flexible and as powerful as those found on the BBC Micro or the Spectrum.

A close look at GET and PUT

Of all the graphics commands on the Dragon, GET and PUT seem to be the most powerful, and the most mysterious. Using GET you can save any rectangular area of the screen in an array and can then use PUT to produce new copies anywhere on the screen. In practice the need for these two commands is less than you might expect. It is often easier and faster to re-draw the shape and then use GET and PUT to reproduce it. However, a little knowledge of how the two commands work leads us on to to ways of using them that the Dragon manual never hints at.

What happens when you use GET depends on whether you end the command with 'G' or not. When using the GET command without 'G' then all that happens is that the display information within the specified rectangle is read into the array memory location by memory location. In other words, GET transfers whole screen memory locations into the array. For example, if you GET a 6 by 6 square into the array A in PMODE 4 using

```
GET (0,0)–(5,5),A
```

then six memory locations within the array will be used. The first will store the screen memory location that holds the top row of six dots within the rectangle, the second will store the screen location that holds the second row of dots and so on down to the sixth row of dots. Now in PMODE 4, or in any two-colour mode, a screen memory location determines the colour of eight dots so although you might think that the GET instruction will only store the information about the six rows of six dots within the rectangle information can actually be stored on six rows of EIGHT dots whether you like it or not!

Screen memory locations

This idea of saving whole screen memory locations is quite a tricky one so it is worth looking at the way the screen memory locations affect what is displayed on the screen. In a two-colour mode each memory location within the screen area determines the colour of eight dots. In the same way, in four-colour mode each memory location controls that colour of four dots on the screen. If you imagine a screen made up of horizontal rows of dots then, in a two-colour mode, the first eight dots in the top row are controlled by the first screen memory location, the next eight to their right by the second memory location and so on to the end of the row. In a four-colour mode the correspondence is the same but with groups of four dots rather than eight. When you move down to the screen to the second row of dots the pattern is repeated, and so on to the very bottom of the screen. To see this process in action in a two-colour mode try the following program:

```
 10  PMODE 4
 20  PCLS
 30  SCREEN 1,1
 40  A=&HFF
 50  B=0
 60  I=0
 70  FOR X=&H600 TO &H600+4*
     &H600 STEP 2
 80  I=I+1
 90  POKE X,A:POKE X+1,B
100  FOR K=1 TO 100:NEXT K
110  IF I=16 THEN I=A:A=B:B=I:I=0
120  NEXT X
130  GOTO 130
```

The program POKEs all ones or all zeros into alternate screen locations so that the group of eight dots that each controls may be seen. You should be able to see that the screen is divided into 32 columns, each eight dots wide. If you want to see the pattern for a four-colour screen then change line 10 in the above program to read

```
10  PMODE 3
```

You will see the same pattern, but each of the lines contains only four screen dots.

These groups of eight or four dots that are controlled by a single memory location are very important to the understanding of the way GET and PUT work. If a screen memory location controls even one of the data within the CRT rectangle then the whole memory location is saved in the array and this means that all the information about the other points that it controls is saved in the array. In this sense the rectangle can be imagined as a sort of frame that is placed on the screen and any screen memory location that is even partially within it is stored in the array.

Inside the array

The screen locations are stored in the array in row order in the same way that the screen memory locations correspond to screen positions. That is, the memory locations from the first row within the rectangle are stored in the array first, working from left to right, then the second row and so on. If GET is used to store graphics information in an array, when you try to examine the contents of the array you will find mainly zeros. The reason for this is that GET completely ignores the way the array, or even standard BASIC numbers, are stored. Both the GET and PUT commands treat an array as a sequence of memory locations and ignore its division into separate numeric elements. So, for example, DIM A(25) reserves 26*5 memory locations because the array has 26 elements and each element (a real number) needs 5 memory locations. Similarly DIM A(9,9) reserves 100*5 memory locations. As far as the GET and PUT commands are concerned, the BASIC arrays are used as areas of storage. No use is made of the fact that the array is one or two dimensional. So, as far as GET and PUT are concerned, DIM A(1,4) and DIM A(9) provide the same amount of storage (ie 50 memory locations) and the screen memory locations are simply stored one after the other in this area of storage.

Now although the Dragon manual suggests that if you want to GET a rectangle with n x m points in it, you need an array DIM A(n,m) this is in fact a gross overprovision of memory. It will always work but it is always too much! If you want to GET a screen rectangle n x m into the smallest array that will hold it, all that needs to be done is to work out how many screen memory locations are within the rectangle and dimension an array that reserves this many memory locations. This is not an easy calculation but even an approximation saves a lot of memory. For example if you are GETting a 5 by 5 rectangle in PMODE 4 the largest number of memory screen memory locations that this involves is ten (ie two screen memory locations per row) so the array can be as small as DIM A(1) ie an array with two elements. If you follow the Dragon manual's recommendation you would have to reserve an array DIM A(5,5) ie 25 array elements or 25*5 memory locations!

GETting in a mess

A consequence of GET storing whole screen memory locations is that PUT restores whole memory locations. This means that if you GET a rectangle that includes part of a screen location when you

PUT it back anywhere else on the screen, all of the memory locations that were stored in the array are restored to the screen, and so an area outside the rectangle that you specified in the PUT is altered. What is worse, PUT restores the screen memory locations in the same way as GET stored them, which means that if the PUT rectangle happens to overlap fewer or more screen memory locations than the GET rectangle did the result will be a confused mess! This fact is usually ignored in descriptions of how GET and PUT work together but if you want to check that it is true try:

```
10  PMODE 4
20  PCLS
30  DIM A(10)
40  SCREEN 1,1
50  LINE (0,0)-(7,5),PSET,BF
60  GET (7,0)-(12,5),A
70  PUT (12,50)-(17,55),A
80  GOTO 80
```

Line 50 draws a small solid square with corners at 0,0 and 7,5. The GET statement should store screen information in A from within a square with corners at 7,0 and 12,5. This GET square only overlaps the solid square in a small vertical strip one dot wide, ie the line from 7,0 to 7,5. (Confusion may be resolved by drawing things out on graph paper.) When the information is PUT back on the screen all one would expect to see is this vertical strip. What in fact can be seen is the whole of the small square reproduced at the new location! The reason for this is, of course, that the GET doesn't store information just from within the area specified, it stores whole screen memory locations. If you think about the memory map for PMODE 4 you will realise that the GET area just overlaps two columns of adjacent screen locations. For example, the first screen memory location determines the colour of the first eight dots in the top left hand corner of the screen and GET area includes the eight dots on the top line, so this whole screen memory location has to be stored along with screen location two, which determines the colour of the next eight dots that also fall within the square (remember that in a two-colour mode a screen location determines the colour of a single row of eight dots).

Full graphics detail

To overcome this problem there is a second form of both the GET and the PUT command. If the GET command is followed with a 'G' then, instead of storing whole memory locations, each dot on the screen is treated separately and the bit or bits that control it are 'packed' into the array. The original groupings into eight or four dots controlled by the same screen memory location are completely ignored. This means that the only graphics data that is stored in the array comes from within the area defined by the GET rectangle. Similarly, the PUT command only alters the area of the screen within it's specified rectangle. To see this add ',G' to the end of the GET command in the last example and ',PSET' to the end of the PUT command and you will see that, instead of a square appearing at the new location, all you get is the vertical line that was predicted! All in all it is easier to predict the result of a GET command ending in 'G' than the result of a GET command that doesn't! For this reason it is better to always use the 'G' unless you have a good reason for not doing so.

The GET command works by storing whole screen memory locations into an array in row order and from left to right. The PUT command simply transfers these memory locations back to the screen where ever they are required.

If the GET command is followed with a G then screen memory locations are ignored and the screen information is stored in the array dot by dot, and no information outside the GET rectangle is stored. In this case the PUT command will not affect anything outside its specified rectangle.

As a side effect the GET and PUT commands are slower when G is specified but you can use additional commands in PUT, such as PSET, NOT etc to determine how the information will be restored to the screen.

Examining arrays

All this is a little complicated to follow as it requires a good understanding of the format that is used to store screen data. If you want to examine the contents of an array that has been used in a GET to check that you understand how the information is stored use –

```
1000  FOR I=VARPTR(A(0)) TO
      VAROTR(A(d))+5
1010  PRINT PEEK(I);",";
1020  NEXT I
```

where d is the maximum dimension of the array A (ie DIM A(d)). This routine will dump the contents of each screen location that forms the array. If you want to know how it works then look up the definition of VARPTR in the Dragon manual.

PUTting user defined graphics

Apart from explaining the odd things that can happen if you use GET without a G at the end, knowing how the GET and PUT commands work seems a little academic. However, now that we know the data format used by the simple GET command we can store data directly into a array to define the shape that would be produced by a PUT. In other words, we can PUT an array that contains graphics data has never been the subject of a GET command. This of course is the key to producing user-defined graphics.

The easiest way of working is to use the GET command without the final 'G' and hence in terms of complete screen memory locations. This is because it is very easy to define the bit pattern of a single memory location and then use PUT to transfer this bit pattern to a screen memory location, and thus to a pattern of dots. To avoid problems with PUT rectangles overlapping different numbers of screen memory locations as it is moved about the screen we also have to restrict ourselves to placing the PUT rectangle over whole screen memory locations. This corresponds to making sure that the top left hand corner of the rectangle is always at an X co-ordinate that is exactly divisible by 8. The simplest way of ensuring that this condition is met is to think of the high resolution screen as being divided up into 16 lines of 32 'characters' in the same way that the text screen is. In this case however each of the 'character locations' is an eight by eight grid of dots that can be the subject of a PUT command. A program to both define a disk and 'bounce' it around the screen is given below:

```
10   PMODE 4
20   PCLS 0
30   SCREEN 1,1
40   DIM A(1;,B(1)
50   V=VARPTR(A(0))
60   POKE V,&H3C
70   POKE V+1,&H7E
80   POKE V+2,&HFF
90   POKE V+3,&HFF
100  POKE V+4,&HFF
110  POKE V+5,&HFF
120  POKE V+6,&H7E
130  POKE V+7,&H3C
140  V=8
150  W=8
160  X=X+V
170  Y=Y+W
180  PUT (X,Y)-(X+7,Y+7),A
190  IF X>=247 OR X =0 THEN
     V=-V
200  IFY>=183 OR Y<=0 THEN
     W=-W
210  PUT (X,Y)-(X+7,Y+7),B
220  GOTO 160
```

Lines 50 to 130 define the dot pattern of the disk by storing the bit pattern for each row into the memory locations reserved by the array. By converting each hex number to binary you will see that each one defines the dot pattern in a row of the disk (ie 0 gives a black dot and 1 gives a white dot). The bit patterns are stored in successive memory locations within the array A, the first location of which is found using the VARPTR function. When A is PUT onto the screen each of its memory locations are transferred to the screen in turn to produce the image of the disk. If you want to see why the PUT rectangle has to be placed exactly over a block of eight screen locations change line 140 to '140 V=7' and watch the chaos that follows.

Even if you haven't followed all of the details of how GET and PUT work you can still use the above method of defining characters directly. It is very fast and particularly useful for converting programs from machines that do have user-defined graphics. In fact this form of Dragon user-defined graphics looks remarkably like the method used by the Spectrum.

MEGACYCAL ROAD TRIAL

Peter Luke took Megacycal's A/D Converter CAL package for a road test and was impressed with the quality of the learning package.

A few years ago the teaching of the principles behind the operation of digital devices often involved a combination of traditional text books with the breadboarding of practical circuits. This latter exercise was, quite rightly, thought to be essential if operation of logic ICs was to be fully understood. But the process of breadboarding a digital circuit, even a simple one, is time consuming and could be expensive if in the hands of the inexperienced student.

The advent of Computer Aided Learning (CAL) packages for use with home microcomputers now offers an attractive alternative to this traditional method of teaching. The Merseyside-based company, Magacycal (a pun that hertz?), have recently expanded their range of CAL software aimed at the student of digital electronics. This review concentrates on one of the new additions to the range, the A to D package designed to teach the principles behind the operation of the most common types of analogue to digital converter.

Analogue to digital converters form the interface between the digital environment of the computer and the analogue signals produced by most types of transducer. They are frequently found in control orientated computer systems but are often little understood by those that use them.

The A to D package provides models of four types of converter; the tracking, successive approximation (SAR), dual slope and parallel (flash) types. The documentation that accompanies the disk-based CAL software is an essential part of the teaching package rather than a mere user manual – even in the world of CAL the printed word is still important.

The software uses the familiar shift/break sequence of the BBC's DFS and when the program is running the user will be presented with a menu that allows any of the four models to be selected.

Before going any further, it should be pointed out that to get maximum benefit from the A to D package the user should be familiar with the nature of the function performed by converters and with binary logic codes. Also assumed is an understanding of the operation of digital counters. These topics are covered in two of the other courses in Magacycal's range of software – Digital Devices and ConFunc (see list at end of this review).

The course begins with an investigation of the operation of the 'tracking analogue to digital converter'. Selecting this option from the main menu will prompt the user to select either a system based on voltage or current comparison. The former choice should be made first as it is easier to understand the mode of operation. The program also allows the number of bits used by the model to be selected. Assuming a four bit model based on voltage comparison is selected, the screen will display an output similar to that shown in **Figure 1.** As with all models in the series, the converter is shown as a block diagram of the major elements that go to make up the device. The logic levels of various internal and external lines are indicated by the colour that any particular line displays, while the value of any quantitive parameter is displayed as a signed number.

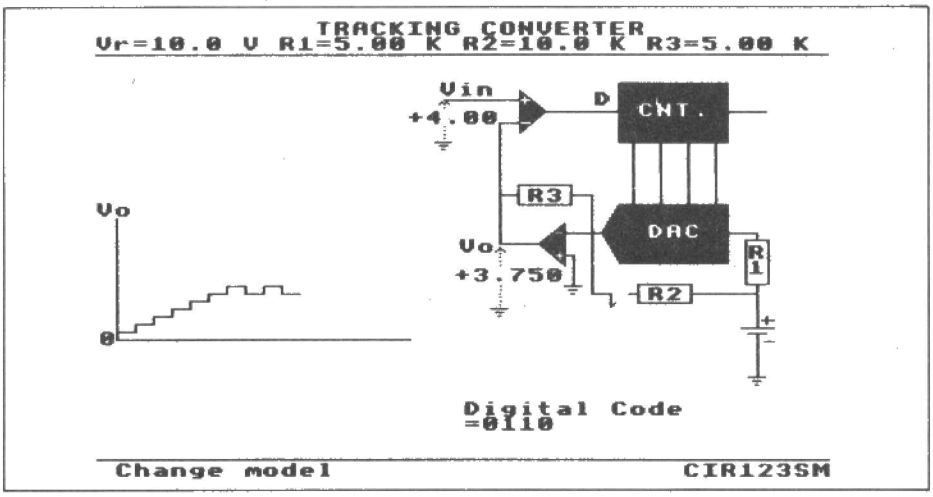


Figure 1. Model of four bit tracking converter.

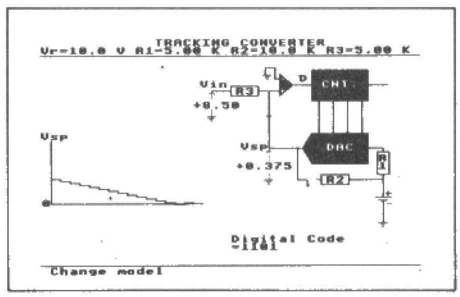


Figure 2. Tracking converter (current based conversion).

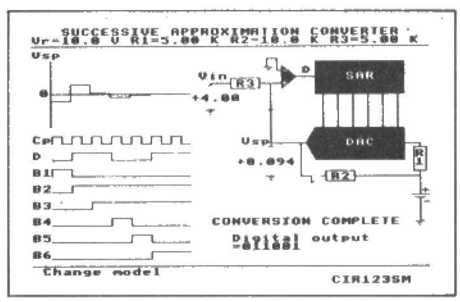


Figure 3. SAR converter with clock waveform displays.

The Manual

The teaching manual begins with a description of the four component blocks that go to make up the converter. As a comparator is an important part of most types of converter, this block is explained in some detail. The operation of this type of converter depends on the loop formed, the DAC (Digital to Analogue Converter), the op-amp and the binary up/down counter.

Modelling of a typical conversion is instigated by simulating the application of a stream of pulses to the counter. This is achieved by selecting the appropriate choice from a menu displayed at the left hand corner of the screen. As with all models, selection of the mode of operation is achieved with the use of just three keys: the cursor shift left, cursor shift right keys and the return key. To apply clock pulses the cursor is first positioned over the C option and the return key pressed. The pulses will cause the counter to count up and thus to increase the current produced

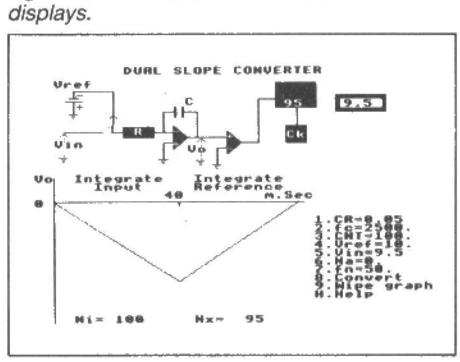


Figure 4. Model of a dual slope converter.

at the output of the DAC that is driven from the four lines of the counter. The current output is converted to a voltage in the op-amp and this voltage is then fed to the inverting (–) input of the comparator. At some stage the value of this voltage will be equal to the external voltage applied to the + input of the converter and will at the next pulse exceed that of the applied voltage. At this point the output of the converter will change and the counter will be put into the 'down' mode. In this way the output of the A to D converter will track the input voltage and with a steady input the LSB of the devices output will oscillate.

The level of the input voltage can be changed by selecting the 1 option from the menu and using the shift left and right keys to alter its value. Hitting return will enter the new voltage value.

The model allows bipolar operation to be modelled. In this mode of operation, the binary value produced by the counter indicates the magnitude of the applied voltage. This is achieved by applying a negative offset voltage to the DAC, the model allows this to be simulated by closing the switch (choose S option from menu).

Having covered the basic operation of the comparator the manual takes the reader through a detailed mathematical analysis of the device. Although it is not necessary to work through this section to understand the operation of the converter, study of this section should be beneficial to most users.

The first section dealing with the tracking regulator concludes with a study of a device based on current rather than voltage comparison. The block diagram of this configuration is shown in **Figure 2** and at a first glance it can be seen that this configuration dispenses with the services of the op-amp. In other respects this configuration operates in much the same way as the voltage based option.

In the last part of this section the relationship between the frequency of the clock signal and the time taken to convert the analogue input voltages into their corresponding digital codes is explained. The maximum rate at which the output of the converter can change (the loop slew rate) is shown to be a function of the full scale voltage of the comparator and the frequency of the clock pulse.

The second section of the manual deals with the principles behind the operation of the successive approximation type of converter. After selecting this from the main menu, the user is once again presented with a number of options concerning the configuration of the device. Assuming the suggested voltage (based), four bit choices are followed, the screen will display the blocks as shown in **Figure 3.**

Operation of this type of converter has much in common with the tracking converter; the major difference is that the up/down counter is replaced by a successive approximation register (SAR). Once again the teaching manual provides a full description of the converter's operation and allows a typical conversion cycle to be modelled by simulating the application of clock pulses to the SAR.

The dual slope converter is introduced in section three of the course. This class of A to D device operates in a very different fashion to the previous two. The central block is an integrator. The device functions by switching the voltage to be converted to the input of the integrator for a specific number of clock periods. This causes the output of the integrator to ramp up to a voltage level. The voltage input under conversion is then removed from the input of the integrator and replaced by a reference voltage of the opposite polarity. This causes the integrator to ramp down back to zero volts. The number of clock cycles taken for the output of the integrator to reach zero will be proportional to the magnitude of the input voltage.

Once again the CAL software allows all aspects of the operation of this type of converter to be studied; a typical screen display is shown in **Figure 4.**

In addition to the mathematical analysis of the operation of the dual slope device the manual explains why this device is able to offer a high degree of noise rejection: that is, the conversion takes place over a period of time allowing any noise signals to be averaged out of the final output.

The final section describes the operation of the flash type converter. This device offers an extremely high speed of conversion because the large number of comparator's. This type of converter can perform the conversion as a parallel process. The high cost of flash converters means that it is unlikely that many people will want to risk such an IC in an experimental environment. So simulation provides a valuable opportunity to study its operation.

At the start of this review a number of the advantages of using CAL packages were explained. The saving in time over traditional breadboarding of experiments and the saving in replacement costs when the inevitable accidents cause damage to devices are just two of these. The Megacycal software offers benefits in addition to this because it allows students to study the internal operation of converters, something that would be very hard to provide with a hardware set up.

When combined with the well written and authoritive manual the clear, on screen schematics make the magacycal package an excellent way to get to grips with the principles of A to D.

Other MegaCycal packages include:

Digital Devices
Confunc
D to A
and D-Logic
MegaCycal, PO Box 6, Birkenhead, Merseyside L43 6XH.

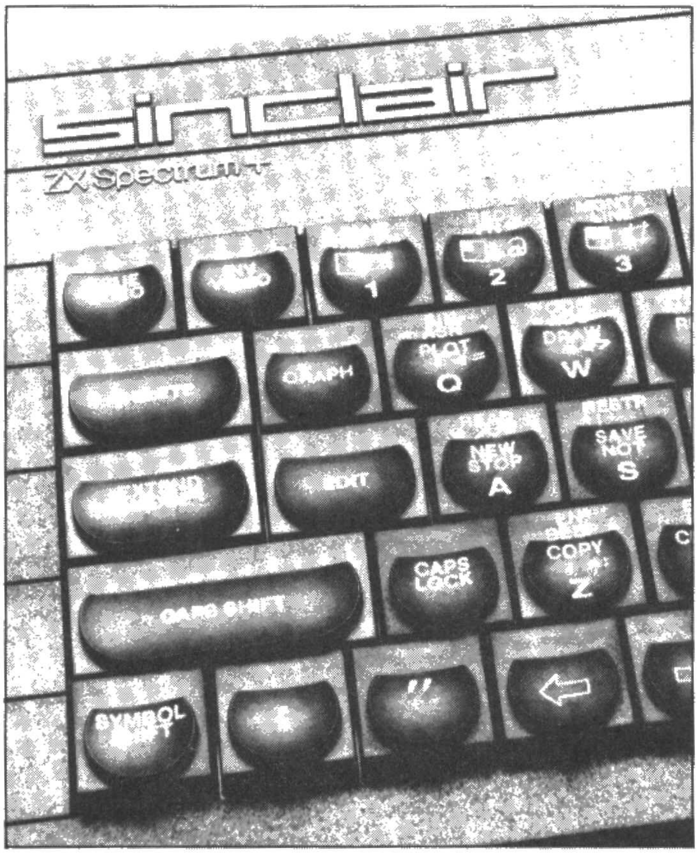
NEW SERIES THE FIRST SPECTRUM WORDPROCESSOR IN FIRMWARE

INTEGRATED WORDPROCESSOR

To celebrate the demise of the rubber-keyboard Spectrum Richard Sargent presents a hardware add-on that turns Sinclair's baby computer into a semi-pro wordprocessor. The series begins next month – this is his brief:

Ask a group of people what makes a good wordprocessor and you're likely to get as many answers as there are people in the group. Here's a selection of possible criteria:

1) A good keyboard.
2) Large capacity memory for long documents.
3) Help messages.
4) The ability to accept fast type speeds.
5) A "What you see is what you get" screen.
6) A mouse or trackball for speedy cursor control.
7) Good printer support.
8) Good mass storage support.



Let's look at the list more closely, bearing in mind the hardware of the Spectrum, and what use its wordprocessor might be put to.

1) The Spectrum now has a tolerable keyboard: rumour has it that some people actually like it. The whole point about keyboards is that the performance is subjective. I liked the old rubber keys of Spectrum Minus, but the fact that the full stop and one or two other critical keys needed to be pressed with the shift ruled it out as a decent wordprocessor. If you are shopping around for an add-on keyboard, bear in mind that the provision of the full-stop on its very own key is no guarantee that it doesn't have to be shifted.

2) Memory size is important. Spectrum user memory starts at 5B00$_H$ (approx), higher if you have Interface One/Microdrives fitted, and higher still if you have Wafadrives fitted. In the worst case, your wordprocessor could start at 63C0$_H$ and finish 4K later at 73C0$_H$. 16K Spectrum users will be distressed about this, since their memory ends at 7FFF$_H$ and they'll only be able to write very short letters on such a wordprocessor. In general though, a 4K or 8K wordprocessor will leave a 48K Spectrum with at least 30K of RAM, and that's enough for 4000 or so words of text.

3) Help messages make deep inroads into free memory space by increasing the size of the wordprocessor. To prevent this, the screenfuls of helpful text are often put on disk (where effectively they waste no space) and down-loaded in the computer when they're needed. Another form of help message is called the user handbook, and this takes up no memory space whatsoever! You may detect that I don't consider help-messages very relevant to the home-computer user, who, after all, should know how his own wordprocessor works, but if the same wordprocessor is being used by school children for example, then the help message is a worthwhile facility and one which should not be omitted.

4) Fast typists can outpace the Spectrum. The reason for this lies in the fact that the computer has a "bit-mapped" screen, and for every letter that is typed (in a wordprocessor application), 6244$_D$ bytes of screen memory have to be loaded with fresh values. There are three ways round this problem. The first is to provide a keyboard buffer, whereby key-presses are collected as a matter of priority, and the screen refresh is done when the pace of typing slows down. The second is to use a smaller screen, for example 12 lines by 32 characters, which takes less time to refresh. The third and by far the simplest option is to ignore the problem on the grounds that 90% of Spectrum users are not professionally-trained touch-typists!

5) "What you see is what you get" is the phrase coined to denote that the screen view of the document is the same as the version which will emerge from the printer, and it's a feature which is only really possible on computers which can deliver an 80 character line to a VDU. The 64 characters-per-line software modification to the Spectrum goes some way to producing the "what you see" effect, but it isn't very readable. The method I prefer is to type the text in 32-character mode, and to switch to 64-characters for previewing purposes, which is how Computer Concept's BBC Wordwise operates.

6) Cursor control can be tiresome on some wordprocessors, particularly if you need to zip all over the document correcting the odd character here and there. Taking the cursor directly to the chosen point with a mouse, or a trackball (which is really an inverted mouse), or a joystick speeds things up no end, and the extra hardware is a useful facility to have in its own right. So the first piece of hardware for the *E&CM* Spectrum wordprocessor is an analogue joystick and the IC to go with it.

7) A proper wordprocessor ought to communicate with a proper printer. Enter hardware item number two, since the Spectrum has no Centronics printer port of its own. The software must be capable of driving the printer, allowing italics, emphasised and double-width print and so on. Double-height characters for Epson-owners is a nice bonus to include, but it all takes up space in the wordprocessor . . .

8) Mass storage for the Spectrum now includes tape, Microdrive, Wafadrives, and floppy disk, and they all have their different LOAD and SAVE syntax. Because of the diversity of storage media, the section of the wordprocessor dealing with LOADs and SAVEs is written in BASIC so that it can be changed to suit individual requirements.

The specification of the hardware wordprocessor should now be taking shape. The minimum configuration consists of printer interface and analogue to digital converter, with some small demonstration routines residing in RAM. The next stage is to add a 4K ROM containing the main wordprocessor (two more ICs), with an option on an 8K ROM for a processor with extended facilities. 4K and 8K ROMs are pin for pin compatible, and can be interchanged provided a 28 pin socket is fitted on the board. The overall system as described above will be housed in a case which will also hold the joystick and four extra switches.

To be continued . . .

PART 2

BBC NETWORKING

Econet is an expensive way of networking BBC micros. Paul Beverley described a low-cost alternative last month. Now he explains how to send messages from satellite computers to the master, and links up two BBC micros to communicate text.

Having found out how to use the BBC micro's RS423 interface to make an ultra cheap network system in which programs were broadcast from a master computer with disk drives, now is the time to reverse the process and send programs back from the satellite computers to the masters.

It is also possible to link up just two BBC micros to send messages from one to the other: this article will show how this can be done, and looks at a few possible applications such as dual computer games.

Sending programs back

Sending programs out from the master to the satellite computers is simplicity itself, but it isn't so easy to send programs back from the individual computers to the master for storage on disk. The problem is that, whereas you can have one RS423 output connected to a number of inputs, it's not possible to connect up more than one output. Each would try to assert a logic level on the data line, and, if they were at different levels, at least one of the ICs would blow.

There are two different solutions to this dilemma, and you can choose which is the most appropriate to your situation.

The first and simplest solution is to use a multi-way switch on the input to the master computer so that it can select which of the outputs from the other computers is actually connected to its input (see **Figure 1**). The disadvantage of this system is the amount of wiring needed. Although you only need a single cable apart from the earth cable to broadcast data from the master, you need one separate wire from each of the satellite computers going back to the master.

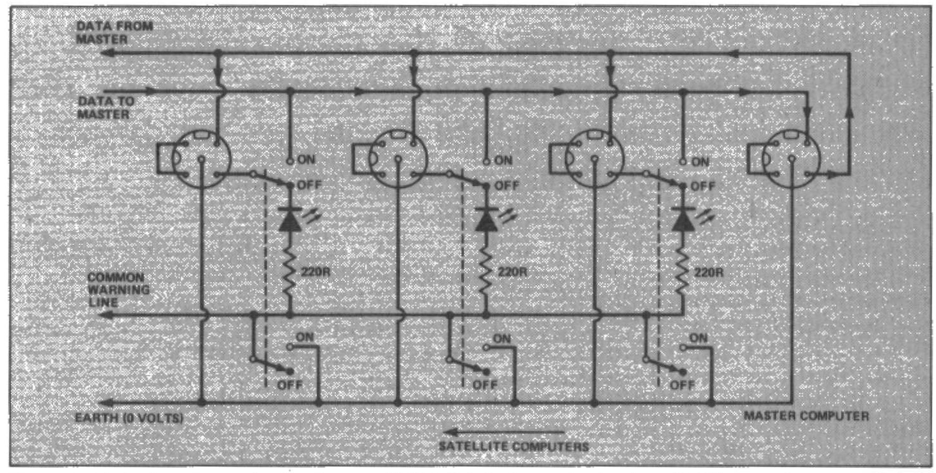


Figure 2. Selecting the outputs of several RS423 interfaces for the input of the master computer.

The second solution is to provide a single common line for carrying the data back to the master. Then you allow the individual computer users to connect their own output onto that data line when they need to send data back. The idea, as shown in **Figure 2,** is to provide a line which links all the satellite computers together and provides a warning facility.

Each computer has a double pole switch so that if the user connects his output to the data input line, it will also connect this common line to the earth line. The effect of this is to cause the LEDs on all the other computers to light up. This is because all their data outputs which, when unused give 5 volts output, will provide current to switch their LEDs on. It is not impossible for two separate computers to connect up to the common data line at the same time, but at least they are given a warning by the fact that their LED is on. You must judge for yourself which of the two systems is more appropriate. Either you can spend more time and money laying plenty of cable and have a fool-proof system, or you can use the cheap and other simple solution which relies on a degree of intelligence in the user.

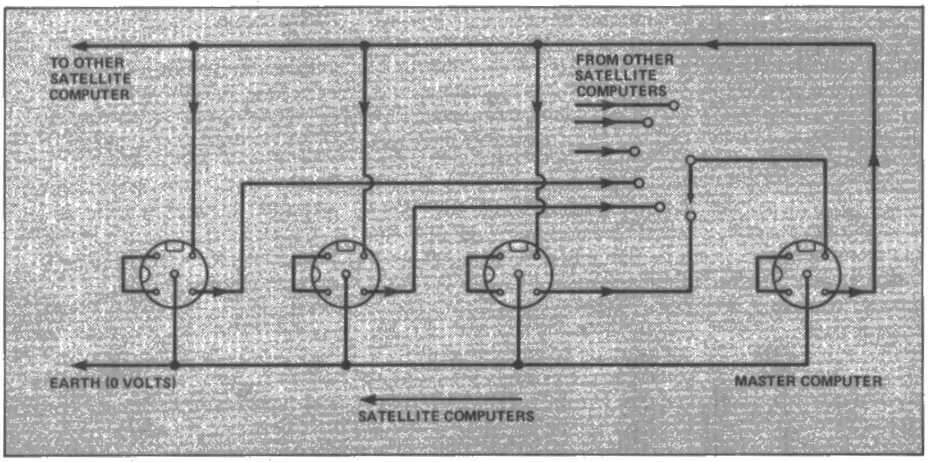


Figure 1. A common line to link all satellite computers and provide a warning system.

Simple RS423 text communication

Turning now from sending programs to sending text, how can two computers be linked together to send messages back and forth? You can start off by making up a simple cable linking the two RS423 connectors. This is shown in **Figure 3a.** All that is needed is two five-way "domino" type DIN connectors and a piece of any three wire cable. It doesn't matter what type is used. If the computers are going to be some distance apart, the cables can easily be long enough to reach right from one end of a house to the other – even the biggest house. If you happen to have two-core screened cable, that will do, since the screen can be used as the ground (0 volts) connection, but it is not really necessary for the cable to be screened. The cheapest, thinnest three-core mains cable will be good enough, but in fact the cheapest way to do it is to buy twin-core speaker flex which is less than half the cost per metre of mains cable – although you have to use two runs it still works out cheaper. One pair for ground plus data transfer is used in one direction and the other for ground again plus data transfer is used in the other direction, as shown in **Figure 3b.**

I have tried using twin-core speaker flex over a distance of 25 metres and there were no errors in the data. Indeed the offi-

cial RS423 specification says that at data rates of under 1,000 baud it is possible to transmit at distances of up to a kilometre, but even this is reckoned to be a conservative estimate and that, at low data rates, distances of several kilometres are possible. It does of course depend to some extent on the type of cable used, but if you use extremely long cables and find errors creeping in, just slow down the data transfer rate to reduce the likelihood of error.

The other electrical connection that has to be made is within the plug itself. The RTS (ready to send) output has to be looped back onto the CTS (clear to send) input. What this does is to assert permanently that it is "clear to send". Without this connection the first few characters that you tried to send would be stored in the output buffer. The program would then halt and the shift-lock and caps-lock lights would both be switched on. The third method of connection, **Figure 3c,** uses the RST and CTS lines for "handshaking" which was explained last month. The RTS line of one computer is connected to the CTS of the other and vice versa. This is useful when data is likely to come so fast that the receiving computer cannot deal with it quickly enough.

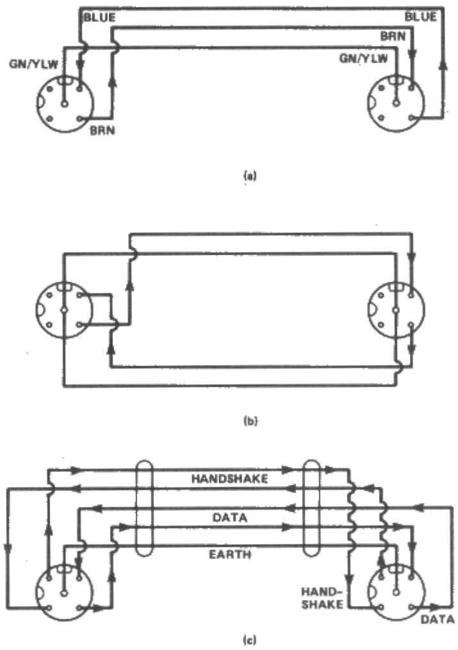


Figure 3. Three ways of making a connecting cable to link two computers via the RS423 serial interface: (a) using a 3 core mians cable; (b) using two twin core cables; (c) using a 5 core (or 4 core plus ground) to add the handshaking facility in order to run at high speeds. The view shown is looking into the back of each of the plugs; the groove being on the left hand side, and the square chopped out slot (if your plug has one) at the top.

The software

Once the two computers are connected up with an RS423 cable, the program shown in **Listing 1** can be used. It allows either of the users to enter messages on his keyboard and these will immediately be sent down to the other computer where they will be displayed. If you are slow at typing and want to have standard words or messages available, the function keys can be used to generate ASCII codes greater than 127 and thus to produce messages. For example:

```
145  IF D%=128 D%=0 : PRINT
     "Hello there!"
```

If the shift key is held down and f0 pressed, the message will be sent. You have to set D%=0 so that the message, but not the 128 code, is sent. If you would prefer to use the function keys without having to press the shift key, adding *FX225,128 as line 305 will make f0 produce ASCII code 128, f1 will produce 129 and so on. To use the copy and cursor keys in the same way put a *FX4,1 command at line 306. The copy and cursor keys will give them ASCII codes 135 to 139. However, it is worth remembering that if you do not do this, the cursor and copy keys can be used in this program to pick up some words off the screen and send them. For example, if your communicant has a word you don't understand you can type in "What does", move the cursor up and copy the word, and then type in, "mean?", to finish off your question.

You can also send control codes. If you think he's gone to sleep send him a ctrl-G, (hold the control key down and press G). This makes his (and your) computer give a bleep. If the screen looks a mess and you want to wipe it clean, use ctrl-L. Provided you are in the same mode then your screen will always be an exact replica of your partners.

Using control with H, I, J and K will move the cursor around the screen so that you can write in a different place and even rub out a word with the delete key or by overtyping with spaces. If you really want to, the mode can be changed by ctrl-V followed by a number, but this is not to be advised as it does not inform BASIC of the new value of HIMEN – the highest memory location it can use without going into the screen memory. This means that if a new mode uses more memory for the screen display than the current mode, BASIC will be using part of the screen memory for its variable storage (the "BASIC stack") and twinkling lights will appear on the screen. If these are over-written the program will probably crash.

A more sophisticated program

Using *KEY command to program the function keys with your own commonly used words and phrases does not work with this simple program, because it goes backwards and forwards between PROCsend and PROCreceive using the INKEY function to pick up a single character at a time for sending and receiving. If you try to use the function keys with this program every other character will be lost as the soft key definition is expanded.

LISTING 1.

```
 10 REM Simple RS423 communication
 20 REM (C) Norwich Computer Services
 30 MODE3 :REM or MODE6 for a TV
 40 PROCinitialise
 50 REPEAT
 60   PROCsend
 70   PROCreceive
 80   UNTIL0
 90 END
100
110 DEF PROCsend
120 *FX3,5
130 *FX2,2
140 D%=INKEY0
150 IF D%>0 VDUD%
160 IF D%=13 VDU10
170 ENDPROC
180
190 DEF PROCreceive
200 *FX2,1
210 *FX3,0
220 D%=INKEY0
230 IF D%>0 VDUD%
240 ENDPROC
250
260 DEF PROCinitialise
270 ON ERROR PROCerror:END
280 VDU19;4;0;
290 *FX7,4
300 *FX8,4
310 ENDPROC
320
330 DEF PROCerror
340 *FX3,0
350 *FX2,0
360 REPORT
370 PRINT" at line ";ERL
380 ENDPROC
```

For more serious applications the ability to prepare text on a disk file ready for sending down to the other computer would be useful. Unfortunately this won't work with the simple program either. If you try it, you will get the same effect as with the function keys, that is, it will only transmit every other character.

The other disadvantage of the first program is that both communicants can add and delete characters at the same time. At first this is fun as you discover that if you are quick you can delete and correct the spelling of the word that the other person has just entered! However, this can soon degenerate into a war to see who can delete what the other person has written faster than he can write it! (To win the war, use *FX12,1 before you start. This speeds up the auto-repeat on your keyboard!)

To overcome such problems the second program (**Listing 2**) was written. This allows only one person to talk at a time and the sender hands over to the receiver by pressing the "]" key (next to the return key). The display shows the text being sent enclosed in these square brackets and the text being received is displayed in curly brackets. When you first run the program it senses whether the other computer is sending and if so, goes into the receive mode. This is indicated to the user by a curly bracket appearing at the top left hand corner of the screen. Otherwise it sends out characters until it gets a reply showing that the other computer is listening. Then it goes into the send mode and an open square bracket appears on the screen to prompt you to start sending.

This program allows you to add lines into the initialisation procedure such as:

```
401  *KEY 0 Hello there!:M
402  *KEY 1 This is Fred speaking.:M
403  *KEY 2 Please could you
404  *KEY 3 Thank you for
etc.
```

Messages can be stored as text files on disk or tape and, for example:

```
175  IF D%=128 D%=0 : *EXEC
     MESSAGE
```

Thus when shift-f0 is pressed, the text file called "MESSAGE" will be taken from the disk or tape and sent down to the other computer automatically. The text files can be prepared using a wordprocessor like Wordwise or, if you have a disk system, by using the *BUILD command. The only thing to watch here is that you do not make the files too long, otherwise, because of the speed at which they are being sent, the receiving computer may not be able to cope and may lose some of the data. If you don't want to send longer files and have problems with lost characters then you can slow down the transmission speed (and the receiving speed at the other end!) Don't forget that the person you are talking to has to have time to read the message, so you can't send too quickly.

Having said that, it is in fact possible to store the received text as it arrives by adding:

```
176  IF D%=129 D%=0 : *SPOOL
     RECtext
177  IF D%=130 D%=0 : *SPOOL
```

Thus, once shift-f1 is pressed all the text subsequently appearing on the screen, both received and sent, would be stored on tape of disk with the file name "RECtext". To stop the storage of text and revert to normal operation, simply press shift-f2 and the file will be closed. If the text is being sent from a disc machine and receiving it on a cassette machine make sure that the sending speed is slow enough not to produce a "log jam" in the receiving computer which would result in loss of characters.

If the RS423 sending speed is slowed down with say *FX8,3 or *FX8,2 and the receiving speed of the other computer similarly with *FX7,3 or *FX7,2, then the characters will be sent so slowly that the receiving computer's input buffer never gets a chance to fill up. What happens instead is that the sending computer's output buffer fills up and no more characterrs are read from the disk until the buffer is partially cleared.

To send information at higher speeds then use the handshake lines, which requires two more lines to be run between the two computers, as shown in **Figure 3c.** When the receiving computer has received so much data that its input buffer is full, it uses its RTS output line to tell the other computer that it is busy. This makes the sending computer wait until the signal on its CTS input line clears.

Applications

The applications of such a link up are mostly on the fun side, but sending text files can have a very practical application. It would be possible to write all sorts of dual computer games – board games like chess or draughts, adventure games where the two of you could venture forth exploring different areas and assisting each other in your quest by sending helpful (or otherwise!) messages back and forth. Perhaps the most obvious game for two computers is battleships; there have already been a couple of listings published in computer magazines. One way of bombing your opponent's target which adds a touch of drama is to use a very slow transmitting speed and move his cursor first across the screen and then down, using VDU 9's and VDU 10's so that he can see the bomb approaching his ships!

Next Month: Extending the idea of text communication to allow it to take place over a standard intercom system or, for the radio hams among you, as a radio signal. The technique used needs absolutely no extra hardware – no modulator, no demodulator – the computer does it all!

LISTING 2.

```
 10 REM RS423 Two Way Communication
 20 REM (C) Norwich Computer Services
 30 MODE3 : REM or MODE 6 for a TV
 40 PROCinitialise
 50 REPEAT
 60   PROCsend
 70   PROCreceive
 80   UNTIL0
 90 END
100
110 DEF PROCsend
120 *FX2,2
130 *FX3,0
140 PRINT"[ ";
150 *FX3,5
160 REPEAT
170   D%=GET
180   IF D%=93 VDU32
190   VDU D%
200   IF D%=13 VDU10,91,32
210   UNTIL D%=93
220 *FX3,0
230 PRINT
240 ENDPROC
250
260 DEF PROCreceive
270 *FX2,1
280 *FX3,0
290 PRINT"{ ";
300 REPEAT
310   D%=GET
320   IF D%=91 D%=123
330   IF D%=93 D%=125
340   IF D%<>126 VDUD%
350   UNTIL D%=125
360 PRINT
370 VDU7
380 ENDPROC
390
400 DEF PROCinitialise
410 ON ERROR PROCerror:END
420 VDU19;4;0;
430 *FX7,4
440 *FX8,4
450 *FX2,1
460 *FX3,7
470 VDU33:REM !
480 IF INKEY2>0 PROCreceive ELSE PROCwakeup
490 ENDPROC
500
510 DEF PROCwakeup
520 REPEAT
530   VDU126
540   W%=INKEY10
550   UNTIL W%>0
560 ENDPROC
570
580 DEF PROCerror
590 *FX3,0
600 *FX2,0
610 REPORT
620 PRINT" at line ";ERL
630 ENDPROC
```

A POCKET GUIDE TO PORTABLES

The lap-held portable micro has been stealthily creeping down in price. For communicating and editing text some portables beat the home micro hands down. William Owen takes a look at the technology that made it all possible.

The computing power contained within a modern lapheld computer would have filled a small room fifteen years ago. Using a number of new, sometimes expensive technologies, all of the functions of a business computer have been squeezed into a box the size of an A4 ring binder.

To be truly portable, a computer must be battery powered, and therefore low powered. The unit must include a screen, a keyboard, a means of mass memory storage, and if possible a printer. Communications facilities are another important requirement in the computer on the move.

Home computer users will know that their computers may look as if they are portable, but meet few if any of the specifications listed above. Most units are mains powered, require CRT monitors, have no integral printer, and have no integral mass storage facilities (Amstrad and QL excepted).

But there is a new family of machines which are truly portable – they at least have screen, keyboard and mass storage unit. The lapheld portables use liquid crystal display, CMOS, thermal printing, and long-life battery technology. It should be noted that at the lower end of the range prices are less than those of some home computers on the market (the NEC 8201A costs only £300). At the top of the market prices go up to £3000 for 16-bit IBM PC emulators with dual internal disc drives and 25 line displays: the technology is such that even this quantity of kit will fit into a small 10lb box.

You can probably imagine the problems faced by the designer of a portable computer. Firstly, cathode ray tubes are bulky and eat up a lot of power – an alternative is needed. Secondly, how do you power a processor and its memory from a battery? Thirdly, what form of mass memory storage can be fitted into a volume of no more than about 14 cubic inches?

Each problem is related to both power and size. The answer to number one is a liquid crystal display (LCD). LCDs are lighter and consume much less power than a CRT, but have much lower visibility and a narrow viewing angle; they also take longer to update. Inevitably an LCD is a compromise solution but the technology has improved lately.

The original LCDs found in calculators and watches were of low resolution, slow, and couldn't stand up to the demands of complex screen addressing. The "twisted nematic" LCDs require the liquid crystal cell to be placed between polarising sheets for the optical change to be observable. This gives an inherently dull appearance and a restricted viewing angle. One answer to the problem is the use of "dye-phase-change" displays, which do not require the use of polarising sheets but instead make use of special dyes which are displayed in the liquid crystal. The end result is a screen with a wide viewing angle, improved visibility, and a higher contrast between the background and the trace over a wide variety of lighting conditions.

Despite other improvements, such as the provision of adjustable angle screens, anyone who has used a portable with liquid crystal display will know that the state-of-the-art is inadequate. This article was typed out on a Sharp PC5000 – it has a relatively good screen but I still had to shine a light onto the display to see the characters. The ultimate answer to the problem will be the electro-luminescent display, such as that used on the GRID super-portable. Electro-luminescence gives a picture as good as that on a CRT but doesn't consume much more power than an LCD. The current drawback, as with all very new technologies, is that it is prohibitively expensive for all but specialist applications.

The solution found to problem number two – powering the chips – is the CMOS low power integrated circuit, which not only consumes less power, but is also faster than the conventional NMOS chips found in most desk-bound computers.

NMOS (Negative Metal Oxide Semiconductor) processors and memory chips require power supplies of +5V or +12V and –12V. CMOS (Complementary Metal Oxide Semiconductor) chips require much lower power levels and compare well with conventional chips in terms of speed. One drawback is that they are expensive, another is that they are delicate and prone to damage by static discharge.

CMOS memory chips will retain their contents with a supply voltage as low as 3V, while drawing a current of only a few nanoamperes (a few billionths of an amp). CMOS is therefore one answer to the problem of portable mass storage: CMOS RAM chips can be used as battery-backed, non-volatile solid RAM disks. Solid RAM acts, as far as the user is concerned, in the same way as a floppy disk drive (except that RAM is faster). The RAM disk is continually refreshed by tiny lithium cells or trickle-charged Nickel Cadmium batteries.

The RAM disk is usually fixed internally and used for temporary storage. However some newer machines already use plug-in battery-backed RAM cartridges which, unlike ROM cartridges, can be both read and written to for permanent storage of data. The NEC-8201A portable can be expanded to 64K internally, and then to 96K through the use of a 32K cartridge containing CMOS RAM backed up by a lithium cell.

Three other mass storage mediums are suitable for use in portable computers, each with a different weight/speed/price ratio. Bubble memory is small and fast but very expensive – a 128K bubble memory cartridge for the Sharp PC5000 is priced at £169 (see 'All about bubbles' *E&CM* October 1984). The cheapest solution is the integral microcassette recorder, chosen by Epson in the HX-20, the first truly portable micro, and by Sord for their IS-11. Users of microcassettes claim that they are reliable and cheap, if a little slow.

The Sony Corporation widened the options when it develsoped the 3.5" disk drive. One lapheld portable, the Data General One, can swallow up two Sony disk drives in its A4 size 12lb frame. The minifloppy requires less care and maintenance than the 5.25" version; it is faster than a microcassette, slower and heavier than bubble memory; it is however a reasonably priced compromise.

There are many solutions to the portability problem and different manufacturers have taken a different path. Most of the machines available are listed in the table, and it can be seen immediately that there are variations in weight, screen size, memory storage, computing power (eight or sixteen bit), interfaces, accessories, and of course software.

Usually the more features a portable has the heavier it is, and this is a point worth bearing in mind. There is a big difference between carrying around a lightweight 4lb

"Home micros have hidden costs – for the screen and mass storage. A portable is an all in one system . . ."

box and a deadweight 12lb micro in a suitcase. Most users will want a screen of 80 characters width, particularly if wordprocessing is to be a frequent application. Depth of screen is less important but seven or eight lines should be considered a practical minimum.

Also, does the manufacturer provide suitable peripherals, in particular, a portable battery-powered printer (essential if you can't wait to get home for the hard copy) and a modem for communicating with your base (or database).

Most portables include bundled software, often in ROM form, and run either CP/M or MS-DOS. However one group of portables, the Tandy, NEC, and Olivetti are based on a non-standard operating system. These machines are of similar design and among the least expensive on

"... but home micros are still a better buy in terms of performance".

the market, ranging in price from £300 to £500. If you just want a cheap portable computer one of them may be the perfect choice for you, but not if you have another computer running CP/M or MS-DOS which you might want to talk to. With WordStar on your desk top micro you will want it on your portable too.

The final choice is between 8-bit and 16-bit computers. As a rule the 16-bit machines are priced at £2000 plus. When you consider that the 32/8 bit QL is priced at £400 (not including monitor) home computers are still a better buy in terms of speed and performance, unless you're after true portability. The one exception to this rule must be the NEC-8201A, at £300. If other manufacturers follow suit anyone who wants a cheap computer for business purposes rather than games may be well advised to buy a portable.

NEC-8201A

PROCESSOR	8085 COMPATIBLE CMOS 80C85
ROM	32K TO 64K
RAM	16K TO 64K PLUS 32K PLUG-IN RAM DISK CARTRIDGES (OPTIONAL)
MASS STORAGE	TAPE CASSETTE (OPTIONAL)
DOS	PROPRIETARY
DISPLAY	40 x 8 CHARACTER LCD, 240 x 64 RES
KEYBOARD	67 KEY, FULL TRAVEL, FIVE PROGRAMMABLE FUNCTION KEYS
DIMENSIONS	295 x 205 x 50mm
WEIGHT	3lbs 4oz
I/O	CENTRONICS, RS232, CASSETTE, BAR CODE READER, 8-PIN AND 9-PIN PHONO SOCKETS
PERIPHERALS	TAPE RECORDER, PRINTER
BASIC SYSTEM PRICE	£300

NEC BUSINESS SYSTEMS 01 267 7000

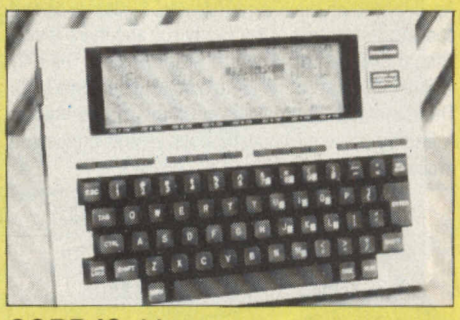
One of three variants built by the Kyocera Corporation of Japan (the other two are the Olivetti and Tandy machines). The 8201A is distinguished by a bigger memory and lower price, and by the fact that it can accommodate self-powered RAM cartridges. It is a simple machine incorporating text editor, operating system, Microsoft Basic and communications programs which enables the user to configure RS232 port for the printer, modem, or another computer. Other applications available include mathematical, financial, wordprocessing and communications software, and of course games. Thoroughly recommended for note-taking and basic wordprocessing – the 8201A is much used by journalists.

TANDY 100

PROCESSOR	80C85
ROM	32K
RAM	8 TO 24K
MASS STORAGE	OPTIONAL TAPE CASSETTE
DOS	PROPRIETARY
DISPLAY	40 x 8 LCD
KEYBOARD	72 FULL TRAVEL KEYS INC. 8 FUNCTION KEYS
DIMENSIONS	295 x 205 x 50mm
WEIGHT	3lbs 4oz
I/O	CASSETTE INTERFACE, BUILT-IN DIRECT CONNECT MODEM, RS232, PARALLEL PRINTER PORT, REAL-TIME CLOCK/CALENDAR, BAR-CODE READER INTERFACE, PIEZOELECTRIC TONE GENERATOR
PERIPHERALS	CONNECTORS FOR PRINTER, PHONE, CASSETTE, ACOUSTIC ADAPTOR FOR MODEM
BASIC SYSTEM PRICE	£449 (8K) £649 (24K)

TANDY CORPORATION – AVAILABLE ALL TANDY STORES

Looks and acts very much like the NEC-8201A but can be expanded to only 32K RAM (from basic 8K). It has a parallel printer interface as well as RS232, and a real-time clock. The bundled software is similar to that available for the NEC model, with TEXT wordprocessor, SCHEDULE record/appointment/address handler and TELECOM communications software. New software includes Scripsit, a fully featured wordprocessor; Expense/Graph Plus (for financial reports and graphics) and Data/Sort Plus, a database manager.

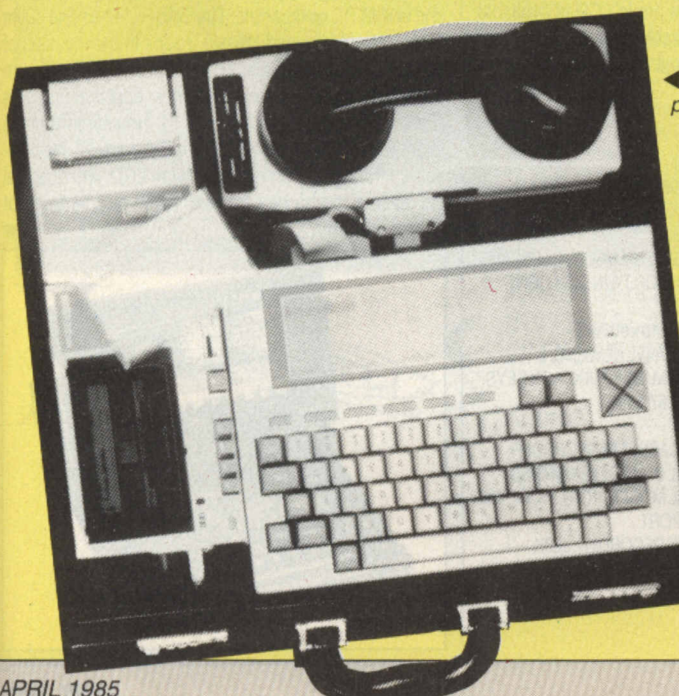


SORD IS-11

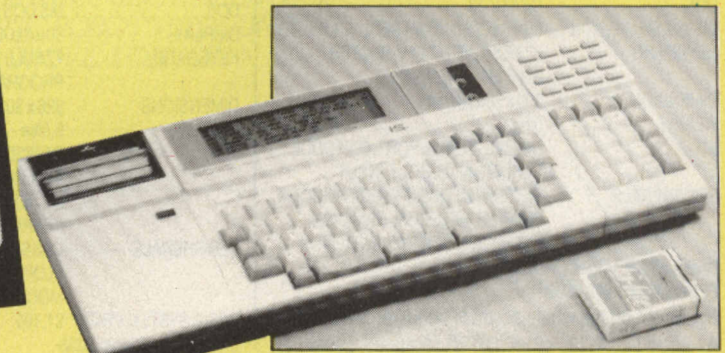
PROCESSOR	Z80 COMPATIBLE CMOS
ROM	64K INC. SORD OS, TEXT EDITOR, SPREADSHEET, DATABASE, COMMS
RAM	32K TO 64K
MASS STORAGE	INTEGRAL 120K MICROCASSETTE
DOS	NON-STANDARD OR CP/M WITH OPTIONAL 3.5 IN. DISK DRIVES
DISPLAY	40 x 8 CHARACTER LCD, 64 x 256 RES
KEYBOARD	72 KEY, FULL TRAVEL, SIX FUNCTION KEYS
DIMENSIONS	300 x 215 x 48mm
WEIGHT	4.4lbs
I/O	ROM CARTRIDGE SLOT, CENTRONICS, RS232C, PARALLEL EXPANSION PORT, BAR CODE READER, NUMERIC KEYPAD PORT
PERIPHERALS	PRINTER, NUMERIC KEYPAD, MODEM, CAR BATTERY ADAPTOR, 3.5 IN. DISK DRIVE, VIDEO INTERFACE
BASIC SYSTEM PRICE	£896

SOCIOUS COMPUTER SYSTEMS 01 631 0787

This machine, although it is slightly more expensive, is probably best compared with the Epson PX-8. The major similarity is that it includes an integral microcassette drive. Like the Epson, the Sord is built around a CMOS Z80, in this case running at 3.4MHz. Limited software is available but the Sord does come with Pips, a powerful spreadsheet/record handling package, and once the promised Sony 3.5" disc drive and interface is available Sord users will have access to CP/M and all its attendant software.



◀ The NEC 8201A portable computer with optional acoustic coupler, printer, and tape recorder. All fit into a suitcase.

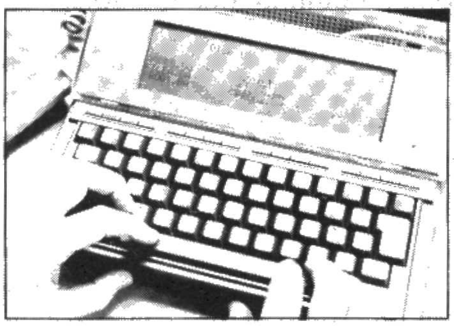


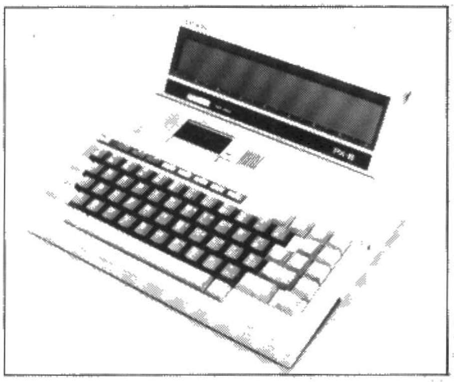
OLIVETTI M-10

PROCESSOR	8085 COMPATIBLE CMOS 80C85
ROM	32K INC MICROSOFT BASIC, TEXT PROCESSOR, ADDRESS LIST, COMMUNICATIONS
RAM	8K TO 32K BATTERY BACKED
MASS STORAGE	OPTIONAL TAPE CASSETTE
DOS	PROPRIETARY
DISPLAY	40 x 8 LCD, 240 x 60 RES
KEYBOARD	57 KEY, FULL TRAVEL, 8 PROGRAMMABLE KEYS
DIMENSIONS	300 x 210 x 50mm
WEIGHT	4lbs
I/O	PARALLEL, RS232C SERIAL, TAPE CASSETTE, BAR CODE READER
PERIPHERALS	ACOUSTIC COUPLER, PRINTER, TAPE RECORDER
BASIC SYSTEM PRICE	£370

BRITISH OLIVETTI 01 785 6666

Again based on the Kyocera design. Similar price and specification to that of the Tandy 100 and there is very little to choose between the two machines. The only visible difference is the M-10's adjustable angle LCD. The standard ROM contains much the same software. A battery-powered acoustic coupler and printer are available. ▼



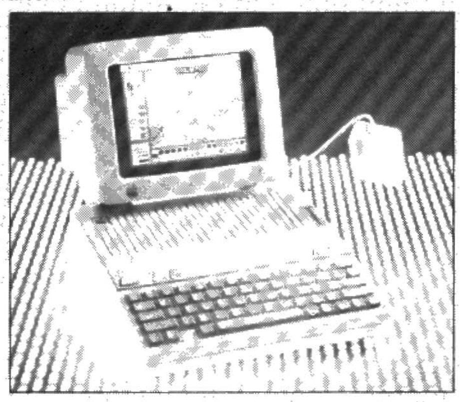


EPSON PX-8

PROCESSOR	Z80 COMPATIBLE CMOS Z84, SLAVE 6301, SUB 750B
ROM	32K WITH CP/M (FOR Z80) AND 4K (FOR 6301)
RAM	64K (FOR Z80) AND 128 BYTES PLUS 4K VIDEO RAM (FOR 6301)
MASS STORAGE	RAM DISK, BUILT-IN MICROCASSETTE DRIVE, EXTERNAL FLOPPY DISK DRIVES
DOS	CP/M 2.22
DISPLAY	80 x 8 LCD, 640 x 64 RES
KEYBOARD	72 KEYS WITH 7 CHARACTER BUFFER, FULL TRAVEL
DIMENSIONS	290 x 205 x 40mm
WEIGHT	5lbs
I/O	RS232C, SERIAL PORT, BAR CODE READER, ANALOGUE INPUT
PERIPHERALS	EXTERNAL LOUDSPEAKER, MODEM, PRINTER, EXTERNAL RAM DISK, EXTERNAL FLOPPY DISK DRIVES
BASIC SYSTEM PRICE	£798

EPSON UK 01 902 8892

Epson set off the whole portable bandwagon with their A4 sized HX-20. The HX-20 suffered from an inadequate 20 x 4 display; this has been replaced in the PX-8 by an 80 column by 8 line version. Unlike many portable manufacturers, Epson have ignored 16-bit technology and instead plumped for a Z80 compatible CMOS chip which runs C/PM. This decision guarantees the user a host of well tried software packages (including WordStar). The inclusion of an integral microcassette recorder has cut down the price of the PX-8 to an affordable level – it must be one of the best buys around.



APPLE IIc

PROCESSOR	6502 COMPATIBLE CMOS
ROM	16K (INC. APPLE BASIC, DISASSEMBLER, MONITOR)
RAM	128K
MASS STORAGE	INTERNAL 143K 5.25 IN. DISK DRIVE
DOS	Pro DOS, DOS 3.3, PASCAL
DISPLAY	80 x 24 LCD (OPTIONAL) CRT MONITOR (OPTIONAL)
KEYBOARD	63 FULL TRAVEL KEYS
DIMENSIONS	305 x 285 x 55mm
WEIGHT	7.5lbs (UP TO 20lbs WITH CRT MONITOR)
I/O	DISK DRIVE INTERFACE, SERIAL PRINTER PORT, MODEM SOCKET, MOUSE SOCKET, TV/RGB SOCKET, VIDEO INTERFACE
PERIPHERALS	CRT DISPLAY, LCD DISPLAY, MOUSE, EXTERNAL DISK DRIVE, PRINTER, JOYSTICK, PLOTTER, POWER SUPPLY
BASIC SYSTEM PRICE	£925 (MONITOR £140)

APPLE COMPUTER (UK) 0442 60244

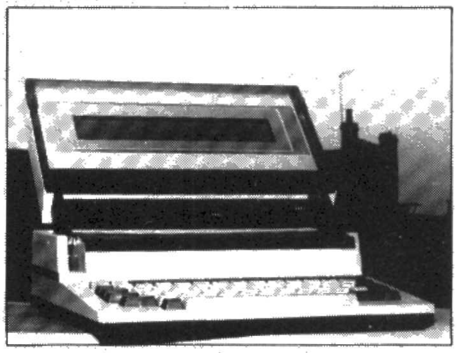
Once upon a time it was a bit of a cheat to call this machine a portable – with the CRT display it weighed over 20lbs. Now an adjustable angle LCD is available and the weight is not appreciably greater than other portables on the market. A must for any mobile Apple enthusiast.

SHARP PC5000

PROCESSOR	16-BIT 8088
ROM	192K
RAM	128–256K
MASS STORAGE	128K BUBBLE MEMORY, CASSETTE RECORDER OR TWIN DISK DRIVE
DOS	MS DOS 2.0
DISPLAY	80 x 8 LCD, 640 x 80 RES
KEYBOARD	72 FULL TRAVEL KEYS INC. 8 PROGRAMMABLE FUNCTION KEYS
DIMENSIONS	326 x 305 x 87.5mm
WEIGHT	9.5lbs
I/O	RS232C, CASSETTE RECORDER, EXTERNAL BUS TO DISK DRIVES, US SERIAL MODEM PORT, NUMERIC KEYPAD PORT
PERIPHERALS	CASSETTE RECORDER, 320K FLOPPY DISK DRIVES, 64K RAM MODULE
BASIC SYSTEM PRICE	£1,195

SHARP UK 061 205 2333

This machine offers an excellent price/performance ratio. The PC5000 makes use of the best of the new low-power technology, including thermal printing, liquid crystal displays, CMOS, and in particular bubble memory, which is available on no other machine in this price bracket. For those who will find bubble cartridges too expensive (at £169 for each 128K cartridge) the alternative options of internal RAM disk, external floppy disk drives, or cassette recorder are available. The integral thermal printer is an important feature which no similar portable can offer; the Sharp may therefore be the only genuine all-in-one portable on the market. ▼

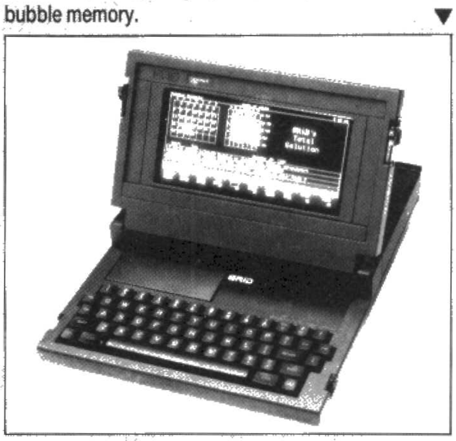


GRID COMPASS

PROCESSOR	TRUE 16-BIT 8086 AND 8087 ARITHMETIC CO-PROCESSOR
RAM	256K EXPANDABLE TO 512K
MASS STORAGE	384K BUBBLE MEMORY AS STANDARD
DOS	MS DOS/GRID OS
DISPLAY	80 x 25 ELECTRO-LUMINESCENT DISPLAY, WITH 320 x 240 BIT-MAPPED ELEMENTS
KEYBOARD	55 FULL TRAVEL KEYS
DIMENSIONS	381 x 292 x 50mm
WEIGHT	10lbs 12oz
I/O	RS232C, RS422, IEEE 488, INTERNAL MODEM
PERIPHERALS	360K PORTABLE FLOPPY DISK DRIVE, COMBINED 360K FLOPPY/10MB HARD DISK
BASIC SYSTEM PRICE	£5195

THAME SYSTEMS 084 421 3471

This is one to ogle at, included here only because it ought to be the shape of things to come. The performance of the electro-luminescent screen can be compared with that of a CRT. The machine has more memory than a standard IBM PC and large and fast memory storage facilities (in bubble), it runs MS DOS and is IBM PC compatible. The Grid PC's has one claim to infamy – it was used by US troops in the invasion of Grenada. Grid offers a special database and software service to users. Other machines approaching this class are the Hewlett Packard 110, Texas Instruments Pro-Lite, and Data General One. Each are 16-bit portables offering an 80 character-wide LCD, with integral 3.5" drives or battery-backed RAM disk – but not bubble memory. ▼



your ROBOT

BRITAIN'S FIRST ROBOTICS MAGAZINE

Each month, Your Robot reports on the latest news from the world of low cost robotics. In addition in-depth features explain the theory behind the operation of robots and computer control systems.

The past month has seen the introduction of a number of robotic devices ranging from the sub £100 Zero 2 from a relatively new company with some old names behind it, to a couple of more expensive products aimed at educational rather than home use.

The Romantic Robot

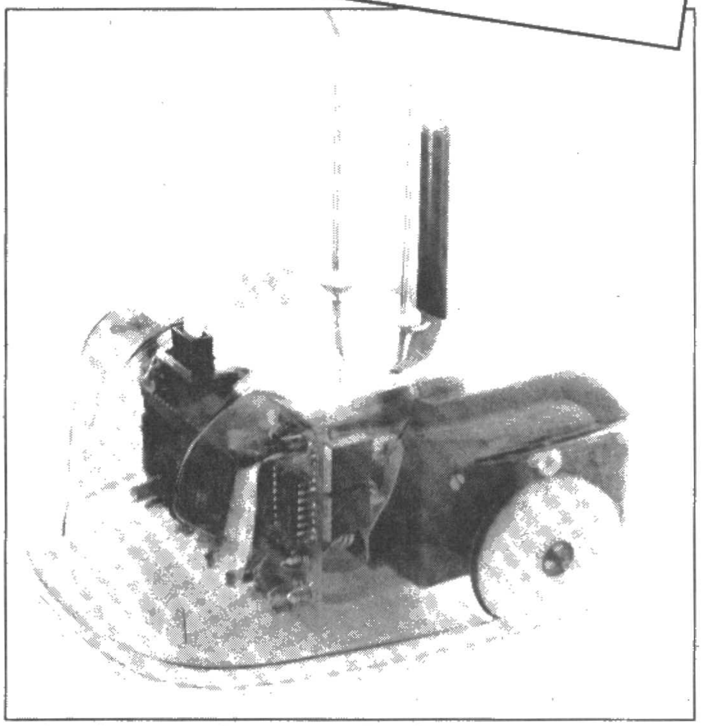
InterGalactic Robots Limited launched their first product, the Zero 2, on St. Valentines Day. With romance in the air the company chose to nickname their turtle-like robot the romantic robot. The Zero 2 is claimed to be the first 'true' robot at under £100 (the robot just scrapes in under this figure costing £99.95 in its assembled form although those prepared to undertake assembly of the device can save themselves £20).

The robot is controlled from the host computer via an umbilical cord down which an RS232 serial data stream controls the three stepper motors that provide Zero 2's motive power. InterGalactic can supply connecting cables for the three most popular school/home micros – the BBC, Commodore 64 and the Spectrum. The latter computer must be equipped with Interface 1 in order to operate the robot.

Zero 2 has two stepper motors which drive the wheels directly. The on-board pen is driven by a third stepper motor and although the accuracy of a stepper motor is not strictly required for this application the robot is designed to the expanded: the company are at present designing a number of add-ons. The body of the robot houses a two-tone horn and a line follower attachment.

IGR have negotiated the right to bundle Sinclair Research Limited's LOGO software with the Zero 2 with an all-in price of £129.95 (this includes a power supply). IGR have developed a unique method of controlling the robot via LOGO. The SPEED and ADJUST commands allow the operation of the robot to be modified for differing applications and enable the user to compensate for any wheel slipage, type wear etc. RPRINT enables the robot to draw – in large letters – whatever is typed out after the command.

IGR have plans for a range of add-on units to the Zero 1. At present they are working on a two way IR link to give greater freedom and a speech synthesis board. Two other planned boards will provide the robot with the means of detecting obstacles, holes or edges, thus avoiding the possibility of falling off the edge of its world.



IGR is a small company with some well known people behind it including both Robin Bradbeer and Dave Buckley. We look forward to bringing **Your Robot** *readers a full review of Zero 2 in the near future.*

IGR, Unit 208, Highbury Workshops, 22 Highbury Grove, London N5 2EE. Telephone 359 2536.

ROB 2

ROB 2 is a sophisticated robot arm that offers a staggering 256 programmable positions within its radius of movement. The arm is driven by stepper motors and offers six degrees of freedom. A major feature of ROB 2 is the comprehensive software that controls the action of the arm. At present this is written for the CBM 64 but versions for the BBC micro and Spectrum should be available in the future.

The various axes of the arm may be controlled directly from the keyboard of the controlling computer which will produce a comprehensive graphic display indicating the exact status of the arm.

We hope to be given the opportunity to put ROB 2 through its paces during the next few weeks and will bring you a comprehensive report in a forthcoming issue of **Your Robot.** *In the meantime those requiring further information should contact Merlin International at 34 Shaftesbury Square, Belfast, BT2 7DB, Northern Ireland.*

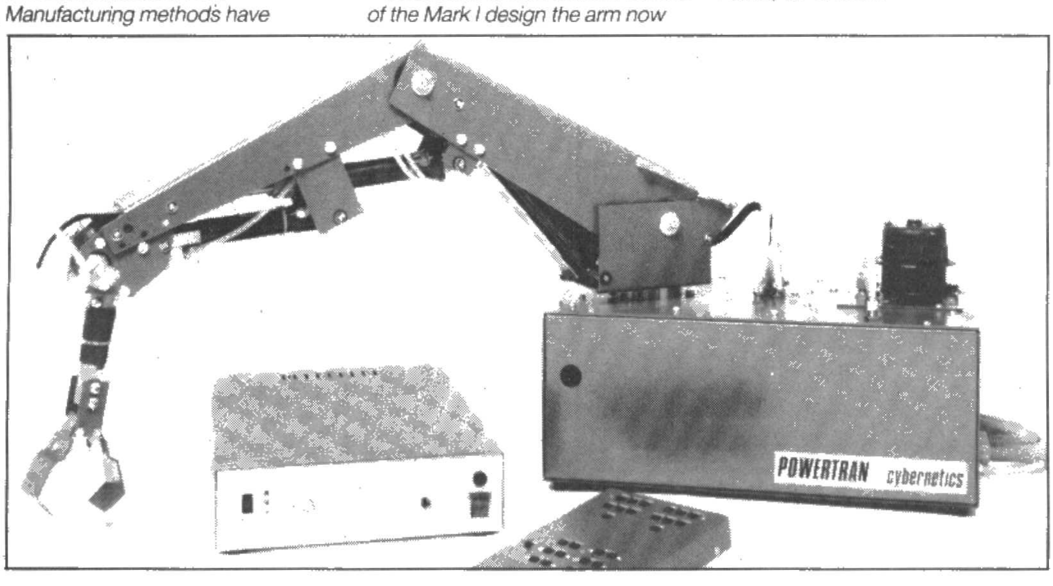
Robot evolution

While the Mark II Genesis P101 and P102 robot arms are not new they do offer a considerable improvement in performance when compared to the Mark I incarnations. New Manufacturing methods have allowed the new designs to offer an increased accuracy of movement while minimizing operational wear and reducing the minotor hydraulic leakages that the earlier design was prone to.

In addition to the RS232 interface of the Mark I design the arm now supports a parallel interface.

Despite these improvements the new version of this popular arm has not increased in price.

Further information – Powertran, Portway Industrial Estate, Andover, Hants, SP10 3NN.



COMMODORE ORATOR

By making maximum use of the facilities offered by the Commodore 64 computer, this speech synthesiser project offers a powerful and flexible system with the minimum of components. Design by A. Dhalla.

Despite extensive coverage of speech synthesis in past issues, *E&CM* have never presented a project for the CBM64. This is rectified below with an allophone based unit built around a low cost, speech generating device which is designed to make construction more simple and less expensive.

Circuit description

The full circuit diagram of the speech synthesiser is shown in **Figure 1.** The low component count of the project is achieved by the use of the General Instruments SPO 256-AL2 Orator IC (see **Figure**

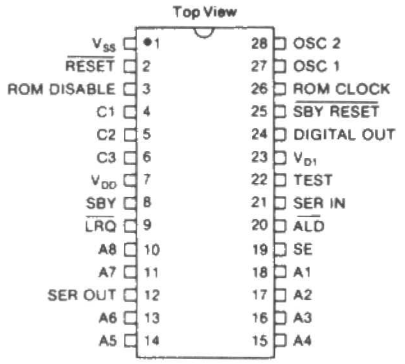


Figure 2. Pin configuration for speech chip.

2 for device pin-out), a VLSI device that copes with the majority of the work associated with the production of speech, and by the fact that the digital filters of the CBM64 computer are used to filter the output of the speech chip to improve the quality of the speech generated. This avoids the use of the analogue filter networks often associated with circuits based on the GI allophone chip. The speech IC incorporates a 2K ROM in which is stored the information required to generate the 64 allophone sounds from which all the synthesised speech is produced. The 64 allophones may be accessed at internal memory locations in the range 0 to 31 (1Fx) – a total of six address lines are therefore required to access all the locations.

When the processor is busy, ie generating speech, the load request pin of the IC (pin 9) is at logic 1. When not busy, or when the internal speech buffer is empty, this pin is at logic 0. To load an allophone address into the internal buffer the address of the required sound is first placed on the IC's address lines (pins 10, 11, 13, 14, 15, 16, 17 and 18). A pulse is then sent to the ALD input (pin 20). At this stage the LRQ pin goes to logic 1 until the internal buffer is once again empty.

A total of eight I/O lines are required to control the speech synthesiser. These comprise seven output lines, six of which are used to form the allophone address, one to provide the active low ALD signal. The eighth I/O line is configured as an input to monitor the condition of the speech IC's busy line. The CBM64 provides an output port than can be configured to meet these requirements. The user port on the left hand side of the computer provides all the necessary signal lines as well as a DC voltage output that can be used to power the speech unit. The pin out of this port is shown in **Figure 3.**

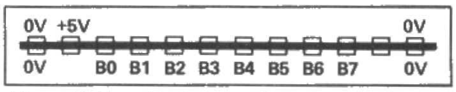


Figure 3. User Port connection.

To configure the port the data direction register of the CBM64, accessed at location 56576 must be loaded with a logic 1 in the bit positions corresponding to the output line and a logic 1 in the bit position that corresponds to the input line. In our case

Parts List

Resistors (all ¼W 5%)

R1	100K
R2	2.2K

Capacitors

C1	0.1μF Disk Ceramic
C2,C3	22pF Ceramic
C4	22nF Polycarbonate

Semiconductors

IC1	SPO 256-AL2
XTL	3.12MHz or 3.27MHz

Miscellaneous

CBM 64 Speech PCB; 28 pin T1 Low profile IC socket; Verobox 301; 2 x 12 way PC 0.156 in Wire Wrap edge connector; 5 pin Din Audio plug.

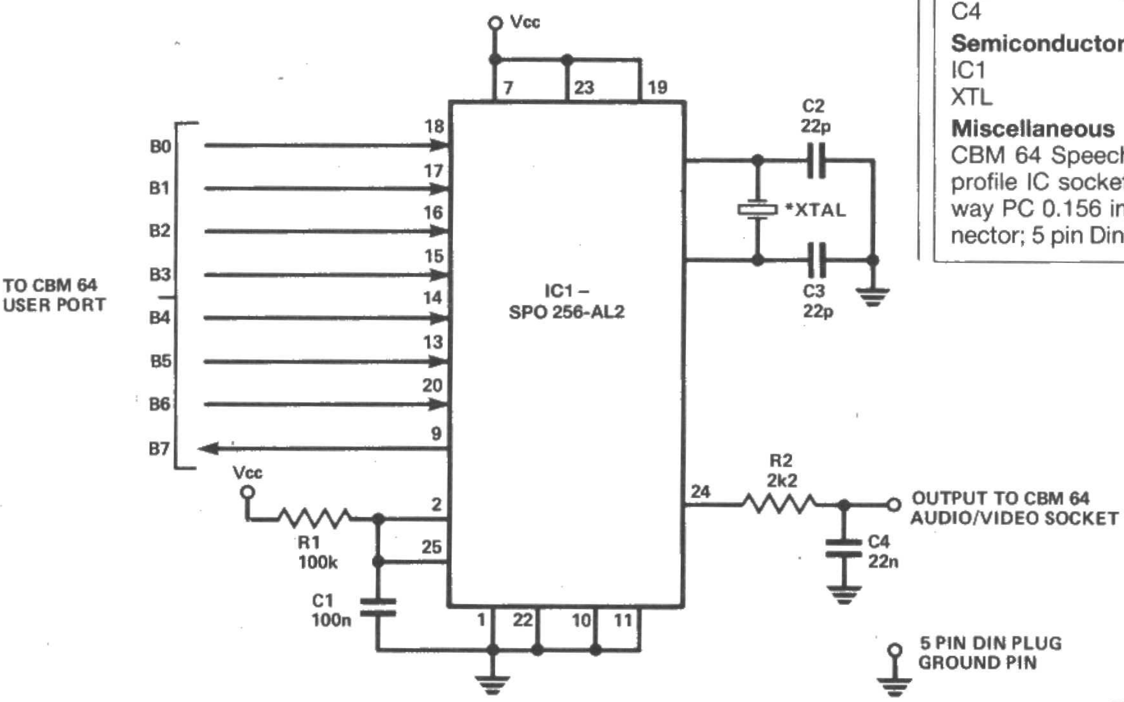


Figure 1. Full circuit diagram.

lines B_0 – B_7 are required as outputs and line B_7 as an input. The value loaded into the DDR is therefore 127 ($7F_H$).

The audio signal at pin 24 is passed through a low pass filter formed by R2 and capacitor C4. The output of this basic filter is fed to the digital filters of the CBM64 via the audio/video socket of the computer. The final output of the speech board is then reproduced via the loudspeaker of the TV set used as the computer's monitor.

The computer's filter can perform high, low and bandpass functions on both internal and external signals. To set a low pass filter bit 4 of memory location 54296 ($D418_H$) must be set to 1 with the remaining bits set to 0. Having determined the filter function, a number of parameters have to be set before using the filter. The first parameter is the break point or cut off frequency of the filter, the value of which is determined by the fact that the frequency span of human speech lies within the range of 0 to 3.5kHz and with this in mind the breakpoint is set at 5.5kHz. The break frequency is set by placing the appropriate values at locations 54293 ($D415_H$) and 54294 ($D416_H$). The second parameter is the filter resonance which produces variations in the speech output depending on its position in the frequency spectrum. For example the sound produced by the letters /z/,/w/,/y/ and /r/ is improved if the filter resonance is positioned near the lower frequencies. The position of the filter resonance is determined by bits 4-7 of memory location 54295 ($D417_H$). The third parameter is the value of the volume level. This is set at the highest available level in order that the final level of sound is under full control of the TV set's volume control. The final parameter that has to be set is the information to indicate to the computer the source of the signal, internal or external. This is accomplished by setting bit 3 of memory location 54295 ($D417_H$) to 1 and bits 2,1 and 0 to 0. Note that bits 7-4 of location 54295 ($D417_H$) determine the position of resonance. More details of the configuration of the CBM64's filter configuration can be found in the computer's programmer's reference manual.

Generating speech

Table 1 shows the allophone set together with the address locations corresponding to the start of each speech sound. To make full use of the opportunities of allophone generated speech it is helpful to have some knowledge of linguistics and an understanding of how words may be broken down into their basic allophone components. We have dealt with this subject in past issues of *E&CM* (see for example September 1984's edition).

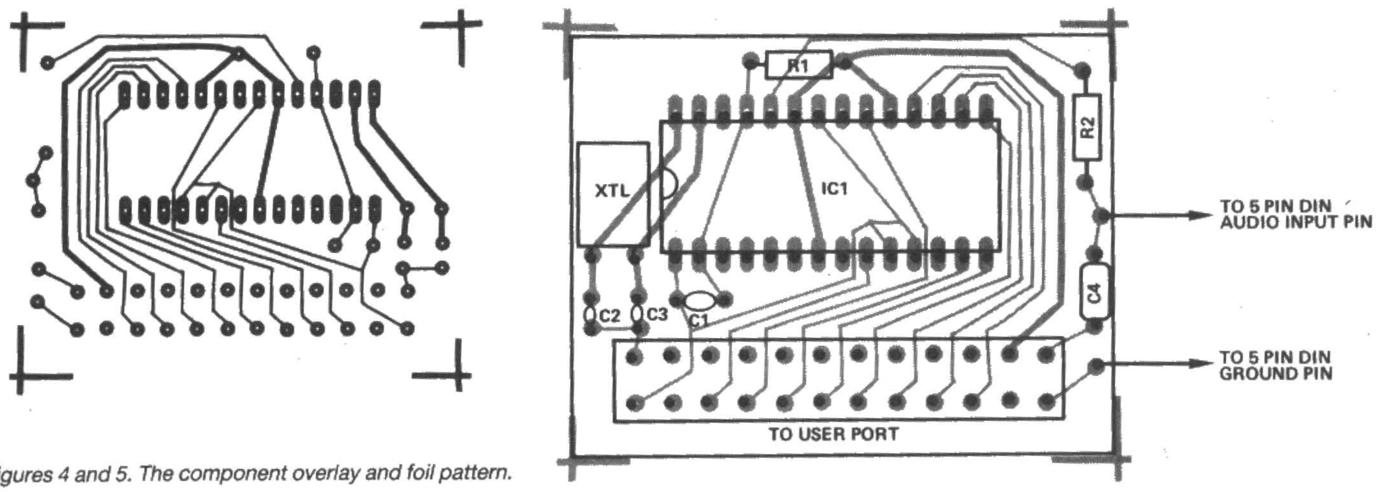
In order to get started however we shall show how the word RATIO may be sounded using the CBM64 speech unit. By making reference to **Table 1** the word is broken down into the component allophones

/RR1/EY/SH/OW/PA2 (14,20,37,53,2).

By loading the SPO 256 AL2's internal buffer with this sequence of →

TABLE 1. The address locations corresponding to the start of each speech sound.

Allophone Addresses		
	Silence	
0	PA1 (10ms)	before BB, DD, GG, and JH
1	PA2 (30ms)	before BB, DD, GG, and JH
2	PA3 (50ms)	before PP, TT, KK, and CH, and between words
3	PA4 (100ms)	between clauses and sentences
4	PA5 (200ms)	between clauses and sentences
	Short Vowels	
12	*/IH/	sitting, stranded
7	*/EH/	extent, gentlemen
26	*/AE/	extract, acting
30	*/UH/	cookie, full
23	*/AO/	talking, song
15	*/AX/	lapel, instruct
24	*/AA/	pottery, cotton
	Long Vowels	
19	/IY/	treat, people, penny
20	/EY/	great, statement, tray
6	/AY/	kite, sky, mighty
5	/OY/	noise, toy, voice
22	/UW1/	after clusters with YY: computer
31	/UW2	in monosyllabic words: two, food
53	/OW/	zone, close, snow
32	/AW/	sound, mouse, down
	R-Colored Vowels	
51	/ER1/	letter, furniture, interrupt
52	/ER2/	monosyllables: bird, fern, burn
58	/OR/	fortune, adorn, store
59	/AR/	farm, alarm, garment
60	/YR/	hear, earring, irresponsible
47	/XR/	hair, declare, stare
	Resonants	
46	/WW/	we, warrant, linguist
14	/RR1/	initial position: read, write, x-ray
39	/RR2/	initial clusters: brown, crane, grease
40	/LL/	like, hello, steel
62	/EL/	little, angle, gentlemen
49	/YY1/	clusters: cute, beauty, computer
25	/YY2/	initial position: yes, yarn, yo-yo
	Voiced Fricatives	
35	/VV/	vest, prove, even
18	/DH1/	word-initial position: this, then, they
54	/DH2/	word-final and between vowels: bathe, bathing
43	/ZZ/	zoo, phase
38	/ZH/	beige, pleasure
	Voiceless Fricatives	
40	*/FF/	These may be doubled for initial position and used singly in final position
29	*/TH/	
55	*/SS/	
37	/SH/	shirt, leash, nation
27	/HH1/	before front vowels: YR, IY, IH, EY, EH, XR, AE
57	/HH2/	before back vowels: UW, UH, OW, OY, AO, OR, AR
48	/WH/	white, whim, twenty
	Voiced Stops	
28	/BB1/	final position: rib; between vowels: fibber; in clusters: bleed, brown
63	/BB2/	initial position before a vowel: beast
21	/DD1/	final position: played, end
33	/DD2/	initial position: down; clusters: drain
36	/GG1/	before high front vowels: YR, IY, IH, EY, EH, XR
61	/GG2/	before high back vowels: UW, UH, OW, OY, AX; and clusters: green, glue
34	/GG3/	before low vowels: AE, AW, AY, AR, AA, AO, OR, ER; in medial clusters: anger; and final position: peg
	Voiceless Stops	
9	/PP/	pleasure, ample, trip
17	/TT1/	final clusters before SS: tests, its
13	/TT2/	all other positions: test, street
42	/KK1/	before front vowels: YR, IY, IH, EY, EH, XR, AY, AE, ER, AX; initial clusters: cute, clown, scream
41	/KK2/	final position: speak; final clusters: task
8	/KK3/	before back vowels: UW, UH, OW, OY, OR, AR, AO; initial clusters: crane, quick, clown, scream
	Affricates	
58	/CH/	church, feature
10	/JH/	judge, injure
	Nasal	
16	/MM/	milk, alarm, ample
11	/NN1/	before front and central vowels: YR, IY, IH, EY, EH, XR, AE, ER, AX, AW, AY, UW; final clusters: earn
56	/NN2/	before back vowels: UH, OW, OY, OR, AR, AA
44	/NG/	string, anger



Figures 4 and 5. The component overlay and foil pattern.

allophone starting addresses the word will be sounded. Note that each word must be terminated with a 'silence' allophone PA1, PA2 or PA3. This is to prevent the final allophone sound from being sounded continuously. The silence periods PA4 and PA5 are used to separate the words of a sentence.

Listing 1 shows a BASIC program that will generate speech from a sequence of numbers written as part of a DATA statement. The required DATA statement is placed at the end of the program as shown. To sound the word 'ratio' it is only necessary to enter the following line as the last line of the program:

10190 DATA 14,20,37,53,2,255,255

Running the program should sound the word. If this test is successful, try altering the value of the cut off frequency of the digital filter by altering the value shown in line 10030 (170). Any number between 0 and 255 may be used. Values close to zero will result in a loss of the /SH/ sound while the value nearest the maximum allowed number will increase the distortion present in the sound. Set the value to that which produces the best results on your TV set.

Altering the value of the filter resonance can also improve the quality of the final sound. This parameter is set by the value given in line 10040 of **Listing 1** and may be in the range 0 to 16 (it is set at 5 in our program). Altering this variable will probably have most effect on the /R/ sound in the word 'ratio'. Again select the value which gives the most pleasing results on your system.

Note that when entering multiple DATA statements with the program shown in the listing, all lines must be terminated with the number 255 with the exception of the last statement which should end with two 255 strings in succession to indicate that no more data follows.

Construction

The component overlay and PCB foil pattern of the speech unit are shown in **Figures 4 and 5.** In view of the low component count construction should pose few, if any, problems. It is recommended that the speech IC should be socketed.

GI recommend a crystal with a characteristic frequency of 3.12MHz for use with their speech IC however these are both expensive and may prove difficult to obtain. The cheaper and more readily available 3.278MHz crystal fortunately gives quite acceptable results. Output is on a slightly higher frequency but the sound is still quite understandable.

Connections to the DIN plug that takes the output of the speech unit to the audio input socket of the computer are shown in **Figure 6.**

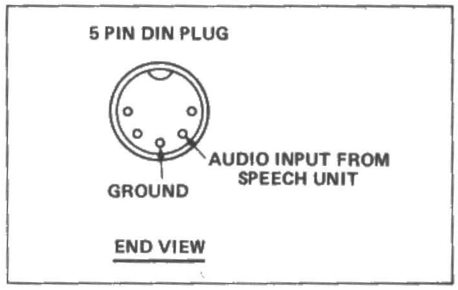


Figure 6. DIN plug connections.

The last component to be fitted is the PCB edge connector.

That's really all there is to it. Construction is made infinitely simpler by the use of the SPO 256-AL2 IC and the quality of sound produced is improved by the use of the Commodore machine's digital filters.

This unit is available in kit form (£14.95 ex VAT) or as a ready built module (£18.75 ex VAT). The machine code routine can be obtained for £2.50. *VLSI Developments, 12 North Parade, Mollison Way, Edgware, Middx. HA8 3QH.*

LISTING 1.

```
10000 RESTORE
10005 REM**SET FILTER AND VOLUME LEVEL*****
10010 POKE (54296). 128+16+15
10015 REM**SET CUT OFF FREQUENCY***********
10020 POKE (54293).2
10030 POKE (54294),170
10035 REM**SET FILTER RESONANCE
10037 REM**AND SOURCE OF SIGNAL************
10040 POKE (54295). 8+(16*5)
10045 REM**LOAD DDR**
10050 POKE (56579).127
10055 REM**SET LDA TO LOGIC 1**
10060 POKE (56577). PEEK (56577) OR 64
10065 REM**********************************
10070 READ A
10080 IF A=255 THEN 10160
10090 B=PEEK (56577) AND 128
10100 IF B>128 THEN 10090
10110 POKE (56577).A
10115 REM**GENERATE NEG. PULSE AT LAD PIN**
10120 POKE (56577). PEEK (56577) AND 191
10130 FOR N=1 TO 7: NEXT
10140 POKE (56577). PEEK (56577) OR 64
10150 GOTO 10070
10160 READ A
10170 IF A<252 THEN 10070
10180 STOP
10190 REM**DATA EXAMPLE********************
10200 DATA 43,19,1,14,1,53,3,255
10210 DATA 46,2,23,44,3,255
10220 DATA 13,3,31,3,255,255
```